

**Script of the  
Training for Iraqi Administrators  
Summer 2010**



Funding:



Deutscher Akademischer Austausch Dienst  
German Academic Exchange Service

Editor: Dr. Nazir Peroz

Authors: Daniel Tippmann  
Daniel Tröder  
Tobias Wölk  
Ralph B. Magnus  
Xun Zhou  
Christoph P. Herbst

Technische Universität Berlin  
Faculty of Electrical Engineering and Computer Science  
Center for International and Intercultural Communication (ZiiK)  
Franklinstr. 28/29, 10587 Berlin  
[www.tu-berlin.de/ziiik](http://www.tu-berlin.de/ziiik)



# Contents

<b>1</b>	<b>Preface</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>A brief history of Unix and Linux</b>	<b>5</b>
3.1	Introduction . . . . .	5
3.2	Early days, 1960-1970 . . . . .	6
3.3	1980's . . . . .	6
3.4	1990's . . . . .	7
3.5	Today . . . . .	7
<b>4</b>	<b>User and Permission Management</b>	<b>9</b>
4.1	Introduction . . . . .	9
4.2	Users . . . . .	9
4.3	Groups . . . . .	11
4.3.1	File and directory access permissions . . . . .	11
4.4	Timestamps . . . . .	14
<b>5</b>	<b>Programs and Processes</b>	<b>15</b>
5.1	What is a program . . . . .	15
5.1.1	Client and Server software . . . . .	16
5.1.2	Execution of programs . . . . .	16
5.2	Processes . . . . .	17
5.2.1	Properties of processes . . . . .	17
5.2.2	Process Management . . . . .	18
<b>6</b>	<b>The Linux Boot Process</b>	<b>21</b>
6.1	General boot process . . . . .	21
6.2	Linux boot loaders . . . . .	21
6.2.1	LILO . . . . .	21
6.2.2	GRUB . . . . .	22
6.2.3	Grub II . . . . .	23
<b>7</b>	<b>The Linux Boot Process – Exercises</b>	<b>25</b>
7.1	Practical Example - Chainloading Grub I and Lilo . . . . .	25
7.1.1	Grub I . . . . .	25
7.1.2	Lilo . . . . .	26
7.2	Boot Loader Backup/Restore . . . . .	27
7.3	Useful commands . . . . .	28
<b>8</b>	<b>The Linux Init Process</b>	<b>29</b>
8.1	Services . . . . .	29
8.2	System V Init . . . . .	29
8.2.1	init scripts . . . . .	30
8.2.2	Runlevel configuration . . . . .	30
8.2.3	Writing init scripts . . . . .	31
8.2.4	Default runlevel . . . . .	32
8.3	Upstart . . . . .	32
8.3.1	Service control . . . . .	32
8.3.2	Service configuration files . . . . .	33
8.3.3	Default runlevel . . . . .	33

<b>9 The Linux Init Process – Exercises</b>	<b>35</b>
9.1 Upstart exercise – Network Monitor . . . . .	35
9.1.1 Task description . . . . .	35
9.1.2 Example output . . . . .	36
9.1.3 Example solution . . . . .	36
<b>10 The Linux Desktop</b>	<b>39</b>
10.1 General overview . . . . .	39
10.2 The X Server . . . . .	39
10.2.1 Special commands . . . . .	40
10.3 Display Manager . . . . .	40
10.4 Window Manager . . . . .	41
10.5 Desktop Environment . . . . .	41
<b>11 Shell scripting</b>	<b>43</b>
11.1 Overview . . . . .	43
11.2 Scripts . . . . .	43
11.3 Variables . . . . .	44
11.4 help . . . . .	45
11.5 Starting Scripts . . . . .	45
11.6 “.” and “..” . . . . .	45
11.7 script arguments . . . . .	46
11.7.1 if and test . . . . .	47
11.8 Redirections . . . . .	49
11.8.1 Input & Output Redirection . . . . .	49
11.8.2 The Pipe . . . . .	50
11.9 Command Substitution . . . . .	51
11.10 Loops . . . . .	51
11.10.1 for loop . . . . .	52
11.11 while loop . . . . .	53
<b>12 Shell scripting - exercises</b>	<b>57</b>
12.1 Overview . . . . .	57
12.2 First Exercise: Directory Watch (***) . . . . .	58
12.3 Second Exercise: File system checker (***) . . . . .	58
12.4 Third Exercise: File Size Analyzer (**) . . . . .	59
12.5 Fourth Exercise: File Checker (**) . . . . .	61
12.6 Fifth Exercise: Batch Rename (*) . . . . .	62
12.7 Sixth Exercise: Font Finder (*) . . . . .	63
12.8 Seventh Exercise: Size Sorter (**) . . . . .	64
12.9 Eighth Exercise: File Finder (**) . . . . .	65
12.10 Ninth Exercise: Find Web Server (****) . . . . .	66
12.11 Tenth Exercise: Archiver (**) . . . . .	67
12.12 Eleventh Exercise: Create svn home directory (*) . . . . .	68
12.13 Twelfth Exercise: Arping all 254 hosts in a given Class C network (*) . . . . .	68
12.14 Thirteenth Exercise: Create a list of files that are opened by a process (****) . . . . .	69
<b>13 C programming</b>	<b>71</b>
13.1 Overview . . . . .	71
13.2 Hello World . . . . .	71
13.3 C preprocessor (cpp) . . . . .	72
13.4 Keywords . . . . .	72
13.5 man pages . . . . .	72

<b>14 Basic Networking</b>	<b>75</b>
14.1 What is Networking? . . . . .	75
14.2 The four layers of TCP/IP . . . . .	76
14.2.1 The Link Layer (lowest) . . . . .	76
14.2.2 The IP Layer . . . . .	77
14.2.3 The Transport layer . . . . .	77
14.2.4 The Application layer (highest) . . . . .	77
14.3 The MAC and the IP(v4) Address . . . . .	77
14.3.1 IANA-reserved private IPv4 network ranges . . . . .	78
14.3.2 There is no place like 127.0.0.1 . . . . .	78
14.4 What Configuration has to be done to get access to the Internet? . . . . .	79
14.4.1 MAC Address . . . . .	79
14.4.2 IP Address/Subnetmask . . . . .	79
14.4.3 (Default) Gateway . . . . .	79
14.4.4 DNS Server . . . . .	79
14.4.5 Manual configuration on the fly . . . . .	79
14.4.6 Manual configuration . . . . .	80
14.4.7 Configuration with DHCP per default . . . . .	80
14.4.8 Illustrated Connection with DHCP . . . . .	82
14.5 Network connection: trouble shooting . . . . .	83
<b>15 Routing</b>	<b>87</b>
15.1 Overview . . . . .	87
15.2 Configuring the kernel routing table . . . . .	87
15.2.1 Showing the kernel routing table . . . . .	88
15.2.2 Adding a route . . . . .	89
15.2.3 Removing a route . . . . .	89
15.3 The default route . . . . .	89
15.4 Behind the scenes . . . . .	90
<b>16 Routing Exercises</b>	<b>95</b>
16.1 Scenario . . . . .	95
16.1.1 Preparing the routing machines . . . . .	96
16.1.2 Configure Network Interfaces: Router 1, Router 2 . . . . .	96
16.1.3 Enabling IP Forwarding for IPv4: Router 1, Router 2 . . . . .	96
16.1.4 Configure Network Interfaces: Router 0 . . . . .	98
16.1.5 Outcome: The configured interfaces . . . . .	98
16.1.6 Outcome: The established routes . . . . .	98
16.1.7 Outcome: Setting up the clients . . . . .	98
16.2 Exercise: Network Redesign (Optional) . . . . .	99
16.3 Exercise: Implementing firewall rules . . . . .	100
16.4 Refine Network Topology . . . . .	101
16.5 Exercise: FW persistence . . . . .	102
16.6 Transfer . . . . .	103

<b>17 Useful administration tools</b>	<b>105</b>
17.1 cron - The Task Scheduling Daemon . . . . .	105
17.1.1 What is cron? . . . . .	105
17.1.2 Different versions of cron . . . . .	107
17.1.3 run-parts . . . . .	109
17.2 date - The System Date and Time . . . . .	109
17.2.1 general syntax . . . . .	109
17.2.2 format sequences . . . . .	109
17.3 find - search for files and directories in the file system . . . . .	110
17.3.1 General syntax . . . . .	110
17.3.2 Advanced options . . . . .	110
<b>18 Secure Remote Administration Tools: ssh and scp</b>	<b>113</b>
18.1 General overview . . . . .	113
18.2 sshd - Secure Shell Deamon . . . . .	114
18.3 ssh Protocol . . . . .	114
18.4 The application of ssh (Secure Shell) . . . . .	114
18.4.1 Typical syntax for remote login . . . . .	114
18.4.2 Typical syntax for remote command execution . . . . .	115
18.4.3 Forward X11 Connections . . . . .	115
18.4.4 Piping through SSH . . . . .	115
18.4.5 Password-less SSH logins . . . . .	116
18.4.6 SSH Port Forwarding . . . . .	117
18.4.7 Other Useful Options . . . . .	119
18.5 scp - Secure Copy . . . . .	119
18.5.1 Options for scp AND cp . . . . .	120
18.5.2 Options only for scp . . . . .	120
18.6 Infrastructure . . . . .	120
18.7 root login via ssh . . . . .	121
<b>19 Partitions, LVM and RAID</b>	<b>123</b>
19.1 File systems . . . . .	124
19.2 Partitions . . . . .	124
19.3 In Practice: Add a partition to the file system tree . . . . .	125
19.3.1 Preliminary considerations . . . . .	125
19.3.2 Check the System . . . . .	125
19.3.3 Create a partition . . . . .	126
19.3.4 Format the new partition . . . . .	127
19.3.5 mount - Integration of File Systems to the File System Tree . . . . .	127
19.3.6 cp -a - Copy Data . . . . .	127
19.3.7 fstab - Fix Position in File System Tree . . . . .	128
19.3.8 Testing and Finishing . . . . .	128
19.4 Logical Volume Manager . . . . .	129
19.4.1 Installation of LVM . . . . .	129
19.4.2 LVM configuration . . . . .	129
19.5 RAID . . . . .	131
19.5.1 RAID levels . . . . .	132
19.5.2 Hardware vs. Software RAID . . . . .	134
19.5.3 Linux Software RAID . . . . .	134

<b>20 LAMP - Linux Apache MySQL PHP</b>	<b>139</b>
20.1 Introduction . . . . .	139
20.2 Apache HTTP Server . . . . .	140
20.2.1 Installation . . . . .	140
20.3 MySQL . . . . .	142
20.3.1 Installation . . . . .	142
20.3.2 Administratering via CLI . . . . .	143
20.3.3 Administering via phpMyAdmin . . . . .	144
20.4 PHP . . . . .	146
20.4.1 Installation . . . . .	146
20.4.2 A Server-side Scripting Language . . . . .	146
20.5 Setting up Web-based open source software with LAMP . . . . .	147
20.5.1 A internal Knowledge Management System . . . . .	147
20.5.2 A Personal Publishing Platform . . . . .	151
20.5.3 A popular Internet Forum Package . . . . .	152
<b>21 Apache Web Server</b>	<b>155</b>
21.1 Apache Configuration . . . . .	155
21.1.1 Structure of configuration files . . . . .	155
21.1.2 Directives . . . . .	157
21.1.3 Virtual Hosts . . . . .	158
21.1.4 Authentication with .htaccess . . . . .	160
21.2 Apache with SSL/TLS . . . . .	161
21.3 User Web Directory . . . . .	166
21.4 Filter for Compression . . . . .	166
21.5 CGI on Apache . . . . .	167
<b>22 MyPHPAdmin: Building a web-based administration interface</b>	<b>169</b>
22.1 The Administration Panel Modules . . . . .	169
22.1.1 A static HTML site with form . . . . .	169
22.1.2 First php site (declaring variables) . . . . .	170
22.1.3 Second php site (working with variables) . . . . .	171
22.1.4 Third php site (hidden fields) . . . . .	172
22.1.5 Fourth php site (including java script) . . . . .	172
22.1.6 Fifth php site (php functions) . . . . .	173
22.1.7 Sixth php site (validating user input) . . . . .	174
22.1.8 Seventh php site (calling external programs) . . . . .	175
22.1.9 Eighth php site (securing cmd calls) . . . . .	176
22.1.10 Ninth php site (the helper class) . . . . .	176
22.1.11 Tenth php site (the navigation) . . . . .	177
22.1.12 Expect - Introduction . . . . .	177
22.1.13 User management with expect . . . . .	178
22.2 CSS Basics (an excursion) . . . . .	180
22.2.1 CSS Syntax . . . . .	180
22.2.2 IDs and Classes as Selectors . . . . .	181

<b>23 Subversion and Trac</b>	<b>183</b>
23.1 Subversion . . . . .	183
23.1.1 Introduction . . . . .	183
23.1.2 Installation . . . . .	184
23.1.3 Getting Started with Subversion . . . . .	184
23.1.4 Subversion Commands . . . . .	188
23.2 Trac and Subversion . . . . .	190
23.2.1 Introduction . . . . .	190
23.2.2 Installation . . . . .	190
23.2.3 Configuring the Environments for Trac . . . . .	191
<b>24 Backup Strategies</b>	<b>197</b>
24.1 Backup strategies . . . . .	197
24.1.1 Why a backup is necessary . . . . .	197
24.1.2 Developing a backup strategy . . . . .	198
24.1.3 using different backups for different data . . . . .	198
24.1.4 Storage Media . . . . .	199
24.1.5 Cycles . . . . .	200
24.2 Backup methods and tools . . . . .	201
24.2.1 File-based backups . . . . .	203
24.2.2 Backing up whole partitions or disks (Images) . . . . .	203
24.2.3 Backup whole file systems . . . . .	207
24.2.4 Backing up databases . . . . .	207
24.2.5 Backup with archiving tools . . . . .	208
24.2.6 Backup with rsync . . . . .	209
24.2.7 Backup with complex large scale software . . . . .	209
24.3 Example scripts . . . . .	210
<b>25 Linux system recovery</b>	<b>213</b>
25.1 root password recovery . . . . .	213
25.2 Boot loader re-writing . . . . .	213
25.3 Simple system backup with tar . . . . .	215
<b>26 Email Server</b>	<b>217</b>
26.1 Introduction . . . . .	217
26.2 Components . . . . .	218
26.2.1 MTA . . . . .	218
26.2.2 LDA, IMAP/POP3 Server . . . . .	219
26.2.3 MUA . . . . .	219
26.3 Installation . . . . .	219
26.3.1 Postfix . . . . .	219
26.3.2 Dovecot . . . . .	221
26.3.3 MySQL . . . . .	223
26.3.4 Dovecot SASL . . . . .	228
26.3.5 PostfixAdmin . . . . .	229

<b>27 Samba</b>	<b>231</b>
27.1 SMB/CIFS . . . . .	231
27.2 An Introduction to Samba . . . . .	232
27.3 Installation of Samba . . . . .	232
27.4 The Samba Server Configuration . . . . .	233
27.4.1 Configuration File Structure . . . . .	234
27.4.2 Most important configuration options . . . . .	235
27.4.3 Testing a configuration . . . . .	238
27.4.4 Creating users . . . . .	238
27.5 Connecting to Samba Shares . . . . .	238
27.6 Printer Shares . . . . .	241
27.7 Summary of Samba Daemon and common used Commands . . . . .	242
<b>28 Primary Domain Controller</b>	<b>245</b>
28.1 What is an authentication server . . . . .	245
28.2 What is a LDAP server . . . . .	246
28.3 LDAP Server Setup - step by step . . . . .	246
28.3.1 Hostname and Domain Membership . . . . .	246
28.3.2 Installation of LDAP server . . . . .	247
28.3.3 Testing LDAP . . . . .	250
28.3.4 Installation of MMC . . . . .	251
28.3.5 phpLDAPAdmin . . . . .	255
28.4 Samba/Windows Users . . . . .	255
28.5 LDAP-authentication and authorization (NSS-LDAP and PAM-LDAP) . . . . .	261
28.6 Mail Users . . . . .	264
28.7 DNS and DHCP Server . . . . .	265
28.8 Linux Clients . . . . .	270
28.8.1 Configure LDAP connections . . . . .	270
28.8.2 Configure PAM . . . . .	272
28.8.3 Setup pam_mount . . . . .	273
28.9 Windows Clients . . . . .	274
<b>29 Proxy Server</b>	<b>275</b>
29.1 Motivation . . . . .	275
29.2 Definition . . . . .	275
29.3 Forward proxy / Reverse Proxy . . . . .	276
29.4 Non-transparent / Transparent Proxy . . . . .	276
29.5 Examples . . . . .	276
29.5.1 Caching debian packages . . . . .	276
29.5.2 Transparent Web proxy . . . . .	278
<b>Bibliography</b>	<b>281</b>



## CHAPTER 1

# Preface

---

The Center for international and intercultural Communication (ZiiK) of the Faculty IV (Electrical Engineering and Informatics) of the TU Berlin is, with financial support from the German Academic Exchange Service (DAAD), active in the reconstruction of the academic structures in the area of Information Technology (IT). Therefore, the team of the ZiiK is creating concepts for improving the IT supply at universities in crisis areas. This includes development and implementation of IT centers and further IT infrastructures for academic institutions in countries like Afghanistan, Iraq and others. Within these projects, a special focus lies on the education and further training of young administrators and lecturers.

In this year also, the ZiiK organized and realized a seven-month training program for 13 lecturers from 13 different universities from Iraq. It was the goal of the program to introduce these participants to technical-administrative processes in order to enable them to manage and expand the IT supply at their home universities.

Besides this technical knowledge, other aspects are crucial for a sustainable IT supply: stable power supply, Internet connectivity, an effective building service engineering for the employment of IT, a solid network infrastructure as well as the demand-oriented creation of computer centers and PC labs. Taking responsibility, understanding IT politics and IT security as well as learning about the role of an administrator belong to this mission of creating sustainable IT infrastructures in Iraq.

The experiences from the past and also this seven-month training program show, that the lecturers, employees and students of the Iraqi universities have an enormous need for education and training in the area of IT. Iraq has the continuing demand for IT education for the development of its academic structures. This means, besides experts like computer scientists, computer engineers and business data processing specialists, also qualified technical personnel like IT managers, digital media designers, IT technicians etc. are urgently needed.

The documentation at hand is the collection of materials that have been trained during this seven-month education program.

For a sustainable education, the participants of this program must have an opportunity to apply their gained knowledge. Therefore, computer centers are to be created at the Iraqi university by them with technical support from the ZiiK. These computer centers are to be the core of the IT supply of the universities.

I would like to thank the Ministry of Higher Education and Scientific Research in Iraq, The Iraqi Embassy in Berlin and the Iraqi universities for the good cooperation. I also thank the German Federal Foreign Office and the DAAD for the financial funding of the project. Last but not least, I would like to thank the lecturers and the team members. Besides the education of the group, a cultural framework program has been offered by the ZiiK. My thanks are also going to the cultural tutors of this program.

Dr. Nazir Peroz, November 2010



# Introduction

---

The contents of this education program were centered mainly on systems and network administration on Open Source platforms, especially Linux. Particular features of the Linux operating system were discussed in detail, also important applications and services. Applied computer networking and the architecture of the Internet, its protocols and network applications has been a part of this.

The purpose of this script is to provide a reference of the topics that were discussed during the lectures and practices of the administrator's training during the seven months. It covers a relatively broad range of materials about Linux, administration tools, applications, computer networking etc. and thus must not be understood as a complete manual for these topics. Rather, it should be used as a compendium to reproduce certain setups and environments. It has been the objective of the course and also this documentation to encourage the participants to self-study and to take the examples and knowledge of this script as and thus create an introductory knowledge about system and network administration.

This text is targeted at users with at least some experience. Basic knowledge about how to use a command shell, navigating through the Linux file system tree, permissions etc. is a prerequisite. Also, knowledge how to use the web as a source of further information and reference is expected, especially for retrieving background information and gaining a deeper understanding of the covered topics. Especially when using this documentation for teaching, further reading is mandatory, as well as the consultation of the man pages of the relevant Linux commands, as they are not always fully introduced or explained throughout the text.

Some of the examples shown here are based on recent Debian/Ubuntu installations (5.0/10.04). When used on other distributions, they might have to be slightly adapted or modified.

A "\$" symbol at the beginning of a line means the prompt of the terminal / command line and *must not* be typed.

Many example commands shown in the text require execution as "root" user (or with `sudo`). It is advised, though, to do this only when it's really necessary and otherwise run them as normal user.

Text in "arrow brackets" ( < > ) is supposed to be replaced in the command line.

We wish you an exciting reading!

The authors



# A brief history of Unix and Linux

---

## Contents

---

<b>3.1</b>	<b>Introduction</b>	5
<b>3.2</b>	<b>Early days, 1960-1970</b>	6
<b>3.3</b>	<b>1980's</b>	6
<b>3.4</b>	<b>1990's</b>	7
<b>3.5</b>	<b>Today</b>	7

---

The history of the UNIX operating system goes back until the early years of computer mainframes, which were used in the 1960's. UNIX has been developed as a multitasking, multi-user operating system from the beginning on. Since then, it has been steadily improved, and forked into a vast number of different "Unix-like" operating systems. Linux is just one of them, and it has the greatest user base and popularity today. Linux and Unix are closely connected to the "Free Software" movement. Thus, many Unix systems are Free Software today, but this has not always been the case.

## 3.1 Introduction

Linux<sup>1</sup>, a flavor of Unix-like operating systems, today is the most widely used operating system of this family. This is for a number of reasons: It is freely available as "Open Source" software and it is designed for security, robustness and reliability especially (but not only) in the area of network services and server systems. It is very flexible and versatile for all kinds of computer use, not necessarily only on desktop PCs, but also mobile devices, cellphones, multimedia devices, cars etc. Its popularity is constantly increasing and today, many big corporations like HP, IBM, Sun, Intel and many more are officially supporting the Linux community and/or even its development.

In this chapter, a brief introduction to the background of this system and its historical background shall be provided.

---

<sup>1</sup>Some people favor the term "GNU/Linux" to emphasize the fact, that Linux is merely an operating system kernel, which in most cases is used in combination with the system tools from the GNU project. In this documentation, for reasons of readability, only "Linux" shall be used. Each individual shall decide by herself, though, which term to choose.

## 3.2 Early days, 1960-1970

The initial version of "Unics" (not yet with an "x") was developed in 1969 by Ken Thompson and Dennis Ritchie at the Bell Labs in the USA. They basically wanted to write a computer game on one of the mainframes (big server-computers at that time) and eventually implemented low-level functionalities, which later led to the development of a whole operating system. The system became very popular inside Bell Labs and soon replaced the formerly used systems. It had multi-user and multitasking abilities very soon and was written in Assembler language. Over the years, it was ported to the C programming language and the name changed to "Unix". In 1975, it was first released outside Bell Labs and was quickly wide-spread, mainly in the academic sector.

Software, at this time, was in most cases delivered as an "appendix" to the expensive hardware, which could only be afforded by big organizations like large companies or universities. There was no particular market of commercial software. Thus, software used to be shared among developers for free, so that everyone could join in the development and submit changes and improvements. Thus, a wide-spread culture of knowledge sharing existed. In 1978, already 600 computers world-wide were running Unix. Big companies like HP, IBM, Siemens and even Microsoft developed their own versions of Unix, all based on the initial code from Bell Labs.

## 3.3 1980's

Computers became smaller and smaller, thus less and less and more powerful, so that more and more people and organizations could afford them. This created a growing customer base for commercial software, and companies started to sell their software products. They commercialized the software market and introduced proprietary software licences, which prohibited any sharing of the programs. From 1981 on, Unix also was sold under a commercial licence which inhibited the free exchange of software among all its users and developers.

This led Richard Stallman, an compassionate Unix user and talented developer, to quit his job at the MIT<sup>2</sup> and to found the "GNU"<sup>3</sup> project<sup>4</sup>. With this project, he aimed at developing a complete operating system which as Free Software to replace Unix. Eventually, he also founded the Free Software Foundation<sup>5</sup> which targets at the promotion and support of Free Software and its developers and the development of software licences, with the GNU GPL (General Public Licence)<sup>6</sup> as the most famous example.

Many other organizations, universities and corporations developed their own flavors of Unix, mostly based on the original Unix code from before 1981. Since then, hundreds of different Unix versions have been created, and many of them still are active and alive and being further developed today: The BSD family (NetBSD, FreeBSD, OpenBSD), Solaris, HP-UX, Irix, MacOSX (Darwin/NextSTEP) to name a few.

---

<sup>2</sup>Massachusetts Institute of Technology

<sup>3</sup>GNU stands for **G**NU's **N**ot Unix, to indicate, that the system is Unix-like, but not the closed source Unix itself

<sup>4</sup>see <http://www.gnu.org>

<sup>5</sup>see <http://www.fsf.org>

<sup>6</sup>see <http://www.gnu.org/licenses/gpl-3.0.html>

### 3.4 1990's

By 1991, most of the unix system tools have been finished by the GNU project, this includes a C compiler, file systems and utilities, shell, text editors etc. However, a working Unix kernel was still missing.

In the early 1990's, a Finnish student at Helsinki University, Linus Torvalds, started to develop some sort of terminal emulation on his new 386 PC, because he was unsatisfied with the current software available. Soon, he added harddisk drivers and low-level hardware I/O functionalities and thus was on the best way to develop an operating system kernel. In 1991, he made version V0.01 of his kernel, which he called "Linux" publicly available on the university's FTP server under the terms of the GNU GPL. From this time on, together with the GNU tools, a usable and free Unix-like operating system was available for everyone.

### 3.5 Today

Since then, Linux (or GNU/Linux) rapidly grew and became more and more publicly used. Many thousand developers all over the world are constantly improving this operating system, many commercial companies are offering services, packages ("distributions"), documentation etc. and Linux today has a growing power and influence of the whole IT world. It is mostly used in the network and server area, where it has its strengths. It is known for its reliability, flexibility and security, and many commercial companies are also officially supporting its use and/or development.

Of all the different Unix versions, Linux is the most popular and most widely used. The term "Unix" is today used not for a particular operating system, but for the whole family of Unix-like operating systems.



# User and Permission Management

---

## Contents

<b>4.1</b>	<b>Introduction</b>	<b>9</b>
<b>4.2</b>	<b>Users</b>	<b>9</b>
<b>4.3</b>	<b>Groups</b>	<b>11</b>
4.3.1	File and directory access permissions	11
<b>4.4</b>	<b>Timestamps</b>	<b>14</b>

---

In the default configuration on most distributions, the Linux user and permission management is relatively simple. System users can be defined with a number of simple attributes and be grouped into system groups. The file permissions are set according to a simple method (read, write and execute for owner, group and others) for each file or folder.

## 4.1 Introduction

## 4.2 Users

A user in a Linux system is an entity in a database. That database entry must consist (at least) of the following entries:

- a **username**
- a **password**
- the users numerical user ID (**uid**)
- the users numerical primary group ID (**gid**)
- an optional description (**GECOS**)
- the users **home** directory
- the users login **shell**

The most common user database on Linux systems is a plain text file ("`/etc/passwd`") that contains a table with one user per line and with columns separated by colons. A single user entry looks like this:

```
daniel:x:1001:1001:Daniel Troeders account:/home/daniel:/bin/bash
```

The password is set to "x". That means, that the password is stored in a separate file "/etc/shadow" for security reasons.

Users can be created, manipulated (modified) and deleted with the following commands: useradd/adduser, usermod, userdel/deluser

To see the options that you can use with these commands, run them with the "-help" argument. For the "add" and "del" commands there are alternatives that do more or less the same, just with a different syntax.

Being a plain text file, the database (" /etc/passwd") can be edited by hand. But making an error can cause the system to become unusable. So please refrain from doing so!

(If you really really must do it, please do only use "vipw". That will start the vim-editor with /etc/passwd opened inside, and will do a syntax check when you save/quit, to at least circumvent syntax errors.)

To print all users known to your system run:

```
# getent passwd
```

This will list users independently of the database they are stored in. (In computer centers it will list users from both: the local machines /etc/passwd file and from a network wide user database.)

Apart from "real users" (humans), there are also lots of "**system users**". They are used to have more fine grained control over the permissions a program can have. System users are not different from normal users - there are places in the filesystem where they can write, and lots of places where they cannot. These users often have as their login shell "/bin/false", or no login shell → both meaning that they are not allowed to login for security reasons.

On most modern Linux systems you will see, that system users are created with uid <1000 and "real" users have uid >=1000.

Nowadays it is common to create a **primary group** for each user, that has the same name as its username. This circumvents common security problems with default group permissions of files and directories.

There is a special user in a Linux system: "**root**". The root user always has the uid "0" and the primary group "root" with gid "0", and can do everything in the system.

```
root:x:0:0:root:/root:/bin/bash
```

Not being restricted by file permissions comes with the risk of easily damaging the system. Working as little as possible as the root user lowers the possibility of an accident. Another problem is, that if a program runs with root-privileges and has a security flaw, an attacker could use that to get control of the hole system.

## 4.3 Groups

A group in a Linux system is an entity in a database. That database entry must consist (at least) of the following entries:

- an **groupname**
- a group **password**
- the numerical group ID (**gid**)
- all the group **member's** usernames, separated by commas.

The most common group database on Linux systems is a plain text file (/etc/group) that contains a table with each line a group and columns separated by colons. A single group entry looks like this:

```
cdrom:x:19:haldaemon,daniel,bacula
```

The groups name is "cdrom", it has its password stored in a separate file "/etc/gshadow" for security reasons, the gid is "19" and it has three members: "haldaemon", "daniel" and "bacula".

Groups can be created, manipulated and deleted with the following commands: groupadd, groupmod and groupdel. To see the options that you can use with these commands, run them with the "-help" argument.

As for /etc/passwd the groups local file /etc/group can be manipulated by hand, but again: don't do it! (And if you *really* must, do use "vigr"!)

To **print all groups** known to your system run:

```
# getent group
```

This will include groups from all active databases.

On most modern Linux systems you will see, that system groups are created with gid <1000 and groups for "real" users have gid >=1000.

### 4.3.1 File and directory access permissions

(Text stolen from <http://www.debian.org/doc/manuals/debian-reference/ch-tutorial.en.html#s-file-perm>) File and directory access permissions are defined separately for the following three categories of affected users:

- the **user** who owns the file (u),

- other users in the **group** which the file belongs to (g), and
- all **other** users (o).

For a file, each corresponding permission allows:

- **read** (r): to examine contents of the file,
- **write** (w): to modify the file, and
- **execute** (x): to run the file as a command.

For a directory, each corresponding permission allows:

- **read** (r): to list contents of the directory,
- **write** (w): to add or remove files in the directory, and
- **execute** (x): to access files in the directory.

Here, execute permission on the directory means not only to allow reading of files in its directory but also to allow viewing their attributes, such as the size and the modification time.

To display permission information (and more) for files and directories, ls is used. See ls(1). When ls invoked with the -l option, it displays the following information in the order given:

- the **type** of file (first character)
  - -: normal file
  - d: directory
  - l: symlink
  - c: character device node
  - b: block device node
  - p: named pipe
  - s: socket
- the file's **access permissions** (the next nine characters, consisting of three characters each for user, group, and other in this order)
- the **number of hard links** to the file
- the name of the **user** who owns the file
- the name of the **group** which the file belongs to
- the **size** of the file in characters (bytes)
- the **date** and time of the file (mtime)
- the **name** of the file.

To change the owner of the file, chown is used from the root account. To change the group of the file, chgrp is used from the file's owner or root account. To change file and directory access permissions, chmod is used from the file's owner or root account. Basic syntax to manipulate foo file is:

```
$ chown newowner foo  
$ chgrp newgroup foo  
$ chmod [ugo]a[+-=][rwx][,...] foo
```

See the man pages of `chown`, `chgrp`, and `chmod` for details.

For example, in order to make a directory tree to be owned by a user foo and shared by a group bar, issue the following commands from the root account:

```
$ cd /some/location/  
$ chown -R foo:bar .  
$ chmod -R ug+rwx,o=rX .
```

There are three more special permission bits:

- **set user ID** (s or S instead of user's x),
- **set group ID** (s or S instead of group's x), and
- **sticky bit** (t or T instead of other's x).

Here the output of `ls -l` for these bits is capitalized if execution bits hidden by these outputs are unset.

Setting **set user ID** on an executable file allows a user to execute the executable file with the owner ID of the file (for example root). Similarly, setting **set group ID** on an executable file allows a user to execute the executable file with the group ID of the file (for example root). Because these settings can cause security risks, enabling them requires extra caution.

Setting set group ID on a directory enables the BSD-like file creation scheme where all files created in the directory belong to the group of the directory.

Setting the **sticky bit** on a directory prevents a file in the directory from being removed by a user who is not the owner of the file. In order to secure the contents of a file in world-writable directories such as `/tmp` or in group-writable directories, one must not only set write permission off for the file but also set the sticky bit on the directory. Otherwise, the file can be removed and a new file can be created with the same name by any user who has write access to the directory.

There is an alternative numeric mode to describe file permissions in `chmod(1)` commands. This numeric mode uses 3 to 4 digit wide octal (radix=8) numbers. Each digit corresponds to:

- 1st optional digit: sum of **set user ID** (=4), **set group ID** (=2), and **sticky bit** (=1)
- 2nd digit: sum of **read** (=4), **write** (=2), and **execute** (=1) permissions for user
- 3rd digit: ditto for **group**
- 4th digit: ditto for **other**

Example:

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Nov  3 23:30 foo
-rw-r--r-- 1 penguin penguin 0 Nov  3 23:30 bar
```

The default file permission mask defines with what permissions files and directories are created by default. It can be set by using the `umask` shell built-in command. See `man 7 builtins`.

## 4.4 Timestamps

There are three types of timestamps for a Linux file:

- **mtime**: the modification time (`ls -l`),
- **ctime**: the status change time (`ls -lc`), and
- **atime**: the last access time (`ls -lu`).

Note that **ctime** is not file creation time.

- **Overwriting a file** will change all of mtime, ctime, and atime of the file.
- **Changing permission or owner of a file** will change ctime and atime of the file.
- **Reading a file** will change atime of the file.

Note that even simply reading a file on the Linux system will normally cause a file write operation to update atime information in the inode. Mounting a filesystem with the `noatime` option will let the system skip this operation and will result in faster file access for the read. See `man mount`.

Use `touch` command to change timestamps of existing files.

# Programs and Processes

---

## Contents

---

<b>5.1 What is a program</b>	<b>15</b>
5.1.1 Client and Server software	16
5.1.2 Execution of programs	16
<b>5.2 Processes</b>	<b>17</b>
5.2.1 Properties of processes	17
5.2.2 Process Management	18

---

Computers are useless without programs to run on them. This chapter gives a brief introduction to programs and processes and how to handle processes graphically and on the command line. It does not cover the operating system details like threads, process creation, scheduling, stack or heap.

## 5.1 What is a program

Without computer programs (or software) computers are useless, thus they breathe life into computers. Programs are sets of instructions for computers. They are nowadays mostly written in high-level programming languages, that are close to human language (and thus relatively easy to understand). Such programs are typically written by professionals in text files, called source code. Computers don't understand those text files. They can only deal with machine language (machine code) which is extremely hard to understand for humans even if we examine very small programs. So the source code must be translated to machine code which is called compilation. This is done either by compilers or by interpreters.

Compilers translate the source code once to machine code which is then stored in executable files. (It takes a while to translate the program, but it has to be done only one time.) Interpreters translate the source code and execute it immediately, step by step. Execution of compiled programs is generally faster than interpretation of source code, but the development cycle edit-compile-run-debug is normally slower than edit-interpret-debug. At the beginning of computer development compiler and interpreter languages were strictly distinguished. Nowadays there are languages that can either be compiled or interpreted. Compiler languages are C, C++, Fortran and Java, to name only a few. Typical interpreter languages are: PHP, Python, Javascript, Perl and Ruby, some of which can be compiled as well.

### 5.1.1 Client and Server software

There are different types of software. Of course there are many different ways to classify software: e.g. the development model and process (closed source vs. open source software), the marketing model (proprietary vs. free software), one can distinguish what the software is used for (e.g. operating systems, development software or pure applications, such as scientific, office, multimedia or educational software) and the complexity of software (complex software suite vs. small scripts).

One very useful distinction is to differentiate between servers and clients. That means that typical tasks of computers are distributed to be taken care of by different programs often on different machines. Communication is often processed by a network of servers that handle the tasks they get from clients, that is from programs controlled by end users. Those servers are typically machines that are running server software to process the requests of the clients in a particular process. Servers are often running on special hardware to process as many as possible of those requests. Thus the separation of tasks leads to a separation of hardware.

For Alice sending an email to Bob, she uses an email client that communicates with a mail server, that is processing her mail to send it to Bob, most likely via other servers.

Client software is designed to "interact" with human beings, using human-interface devices on the one hand, and to communicate with servers on the other hand. Therefore it has often a graphical user interface (GUI). To communicate with servers the client implements well defined protocols, e.g. SMTP for sending email, IMAP for pulling email to a client or HTTP for websites.

Server software is mostly running as a non-interactive background job, waiting for requests, i.e. "listening" on special ports to process requests. Typically those servers (the term server is used for both the software and the actual hardware it is running on) are running 24/7 all week long and are using special hardware and uninterruptible power supplies, to ensure interruption-free operation. Client hardware should only be running when a person is using application software, and is therefore less enduring and correspondingly cheaper.

There are two different sorts of client-server concepts. One is on software (program) level and the other is on hardware level. This section explains the software level (client programs - server programs). The client - server concept describes the relationship of two computer programs. They communicate using well defined protocols, often run on different computers that are connected via network.

Typical client applications are: Firefox as web browser, Thunderbird as email client, Pidgin as instant messenger. The respective servers for these purposes are: Apache as web server, Postfix as SMTP server, Dovecot as IMAP server and Jabberd as XMPP server.

### 5.1.2 Execution of programs

In Linux almost everything is a file. So executable programs are special files: They are either in a machine code binary format or source code in an interpreter language. Not every binary can run on every operating system or every computer architecture. It has to be compiled according to the hardware and the OS.

To run a program, its program file has to be executed. In the permission concept of Linux, there is special bit (the execution bit) that has to be set in order to allow a user or a group to run a program (see also chapter 4).

Programs can be invoked by special scripts (init scripts), or by a running another program to interpret the script code.

## 5.2 Processes

A program itself is just a set of instructions, while a process is the actual execution of those instructions. Unix-like operating systems are designed as "multi-tasking" operating systems, as the most modern OSes are. The OS itself is complex enough to have always a couple of processes running. But as the CPU (or each CPU core) is only able to execute one process at a time, the OS has to manage the execution of all the processes. This is done by a very fast switching between processes and distributing the available CPU working time among them. Thus, the CPU is working on a process only for a very short time until it is switched by the OS (the "scheduler") to the next process.

Several processes may be associated with the same program; for example, opening up several instances of the same program often means more than one process is being executed.

For security and reliability reasons most modern operating systems prevent direct communication between "independent" processes, providing strictly mediated and controlled inter-process communication functionality.

### 5.2.1 Properties of processes

The operating system manages all processes: It assigns Memory, handles the scheduling, assigns a priority (PR), assigns a user (USER) by the user id (UID), a group (GROUP) by the group id (GID) assigns a process id (PID) and the parent process id (PPID) (every process is started by an other process [except the first process, called init]). All those information is stored in a data structure called Process Descriptor.

#### Niceness (NI)

An important property of processes is its "niceness". It determines the scheduling priority in the kernel. A process is said to be the nicer the lower its scheduling priority is.

priority (nice level)	description
19	very nice / lowest priority
0	normal nice / default value
-20	absolutely not nice / highest priority

`nice` runs a command with modified scheduling priority.

```
\$ nice -n <nice level> <command>
```

Only the root user can apply nice levels lower than 0.

### Priority (PR)

The priority value is calculated from values like used CPU time, process type (real time / daemon / interactive), waiting for I/O (input/output) and nice value. Processes with a higher priority get to run more often and longer.

## 5.2.2 Process Management

### Signals

Signals are used to control processes, e.g. by using the `kill` command. Some useful examples are in the below table, for more signals look at the man page of `kill` (`man kill`).

sigName	sigId	purpose
KILL	-9	Destroy a process: erase it from memory (RAM).
TERM	-15	Terminate a process: ask the process for termination (the process terminates itself). This is the default value.
CONT	-18	Continue a process: a stopped process is allowed to be shifted to CPU.
STOP	-19	Stop a process: the process won't be shifted to CPU until continued.

### kill - send signal to process

Send a signal to a process:

```
$ kill <sigId> <pid>
```

```
$ kill -s <sigName> <pid>
```

Send a signal to all processes with the same command name:

```
$ killall <sigId> <processname>
```

```
$ killall -s <sigName> <processname>
```

```
renice
```

Assign a new nice level to a already running process as follows:

```
$ renice <nice value> <pid>
```

### ps - processes

The shell command **ps** reports a snapshot of the current processes. It has a lot of options to display various categories of processes like all processes of one user:

```
$ ps -u <username>
```

List all processes:

```
$ ps -e
```

List all processes with extra full format:

```
$ ps -eF
```

List declared fields of processes ("list of fields" is a comma separated list of process properties written in small letters):

```
$ ps -o <list of fields>
```

List processes in a tree view:

```
$ pstree
```

### top - table of processes

**top** is a interactive program that is controlled by typing keys (letters). By default, it shows a table of all processes ordered by percentage of CPU usage and is updated every 3 seconds.

```
$ top
```

key	command description
? / h	display help (all available commands / keys)
q	quit top
u	show only processes of <user> (not giving a user will display all processes)
k	send signal to a process (PID and sigId must be declared)
r	renice process (PID and NI must be declared)
s	set update interval in seconds
O	set select sort field

`top` also shows important information about memory (RAM) and CPU usage in its head and can help troubleshooting, e.g. if the system became slow. For instance, it shows processes that are consuming an unusual amount of memory or CPU time. This normally means that the process is not working properly and better be terminated or killed, thus the other processes can run properly.

### Graphical tools for Process Management

In Ubuntu and Kubuntu there is a graphical tool for process management called "system monitor". In Ubuntu, i.e. in Gnome, it is located in *System->Administration->System Monitor* in the *Processes* Tab and in Kubuntu, i.e. in KDE, in *System->System Monitor*. In Windows, processes can be managed using the Task Manager, which can be found by right-clicking the task bar and then clicking *Task Manager* or by pressing CTRL-ALT-DELETE and then clicking *Task Manager*.

# The Linux Boot Process

---

## Contents

<b>6.1 General boot process . . . . .</b>	<b>21</b>
<b>6.2 Linux boot loaders . . . . .</b>	<b>21</b>
6.2.1 LILO . . . . .	21
6.2.2 GRUB . . . . .	22
6.2.3 Grub II . . . . .	23

---

This chapter explains the Linux boot process from powering on the machine until the presence of a graphical login screen. It will also give a brief overview about the different boot loaders that can be used to boot a Linux system.

## 6.1 General boot process

Figure 6.1 shows the general boot process. After the system has been powered on, the BIOS will execute any boot loader that is installed in the MBR (Master-Boot-Record) of the configured hard drive. This primary boot loader now has two choices, denoted by A and B. Path A shows the case where the primary boot loader loads a secondary boot loader which is located in the boot sector of any of the partitions of the hard drive. The secondary boot loader eventually gives execution to an operating system kernel. The primary boot loader can also directly load an operating system kernel as shown in path B.

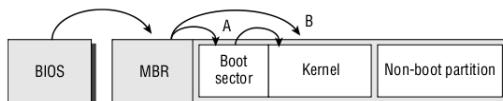


Figure 6.1: General boot process.

## 6.2 Linux boot loaders

Linux has several boot loaders, the most popular are LILO, GRUB and GRUB2.

### 6.2.1 LILO

- LILO - Linux Loader

- Was once the default boot loader for Linux.
- Very small
- Modifications to configuration file make re-installation necessary.
- Configuration file: /etc/lilo.conf :

```
# lilo.conf
#
boot=/dev/sda1      # install LILO into the first partition of the first hard
                     # drive
timeout=150
root=/dev/sda5      # default root device for 'image' entries
#
# Entry to boot linux:
image=/boot/vmlinuz-2.6.25
label=Ubuntu
initrd=/boot/initrd.2.6.25
root=/dev/sda2      # overwrites the global 'root' option
```

- Documentation:

```
$ man lilo
$ man lilo.conf
```

- Installation:

```
$ sudo aptitude install lilo
$ # modify /etc/lilo.conf by hand
$ lilo # installs LILO into the boot sector specified in /etc/lilo.conf
```

### 6.2.2 GRUB

- Widely used and the successor of LILO.
- Modifications to configuration file do not make re-installation necessary.
- Configuration file: /boot/grub/menu.lst
- Examples:

**hiddenmenu** Hides the menu on boot. Can be shown by pressing ESC  
**timeout 120** Wait for 120 seconds until booting the default entry

- Documentation:

```
$ man grub
```

- Installation:

```
$ sudo aptitude install grub
$ sudo update-grub
$ sudo install-grub /dev/sda
```

- Provides shell like boot prompt
- Example:

```
grub> root (hd0,0) # sets grub installation directory
grub> kernel /boot/vmlinuz-2.6... root=/dev/sda1 # specifies the kernel to
       load and gives root partition
grub> initrd /boot/initrd-2.6... # specifies initial ram disk
grub> boot           # boots the kernel
```

### 6.2.3 Grub II

Nowadays, Linux distributions, especially Ubuntu are switching from Grub I to Grub II. Development of Grub I has been discontinued and it is now deprecated. Grub II is a complete rewrite of Grub I which uses a more modular architecture and provides several features over Grub I, e.g. internationalization. It supports more file systems and system architectures than Grub I and is more customizable through a script like language. Additionally, at least on distributions like Ubuntu and Debian, it provides a collection of user-land tools that automatically generate configuration files and probe for installed operating systems.

The following table summarizes some of the differences between Grub I and II.

	Configuration file	Partition numbering
Grub I	/boot/grub/menu.lst	starts at 0
Grub II	/boot/grub/grub.cfg	starts at 1

In Ubuntu the Grub II configuration file is automatically generated by scripts located in the directory `/etc/grub.d/`. The following command will execute the scripts and redirect their output into the file `/etc/grub/grub.cfg`:

```
$ sudo update-grub
```

As the Grub II configuration file is auto-generated, customizations should never be done at this file directly, because they will be overridden when re-running `update-grub`. Instead, there is a special file, `/etc/grub.d/40_custom` that can be used to add entries to the Grub II boot menu. Below is a listing of that file which adds an entry to chain load into Lilo in the same way as Grub I.

```
# /etc/grub.d/40_custom
#
menuentry "LILO" {
    set root=(hd0,1)
    chainloader +1
}
```

The commands are similar to that of Grub I. Instead of *title*, *menuentry* will start a new menu entry. The location of the boot loader that should be chain loaded is specified with *set root=(hd0,1)*. Notice the different partition numbering than in Grub I. After modifying any of the files in */etc/grub.d/*, the configuration file must be re-generated with:

```
sudo update-grub
```

For a more complete documentation of Grub II look at <http://grub.enbug.org/Manual>.

# The Linux Boot Process – Exercises

---

## Contents

<b>7.1 Practical Example - Chainloading Grub I and Lilo . . . . .</b>	<b>25</b>
7.1.1 Grub I . . . . .	25
7.1.2 Lilo . . . . .	26
<b>7.2 Boot Loader Backup/Restore . . . . .</b>	<b>27</b>
<b>7.3 Useful commands . . . . .</b>	<b>28</b>

---

This chapter is a continuation of the Linux Boot Process chapter. It gives an example on how to install Grub I and Lilo, so that it's possible to chainload one from the other. It presents an easy way to backup and restore a boot loader with Linux on-board equipment.

## 7.1 Practical Example - Chainloading Grub I and Lilo

The example discussed in this chapter, shows how to install both, Grub I and Lilo. It will present how to add the necessary menu entries to chainload each one from the other. In this example, Grub I will be the primary and Lilo the secondary boot loader. That means, Grub is installed into the MBR of the first hard disk and Lilo into the boot sector of the first partition of that hard disk.

### 7.1.1 Grub I

Below is a listing of the Grub configuration file `/boot/grub/menu.lst`, used in this example. It contains one entry to boot a Linux kernel. The other entry is used to chainload into the Lilo boot loader.

```
# /boot/grub/menu.lst
#
# Global Options
#
default 0
timeout=150
#
# Entry to boot Linux:
#
```

```

title Ubuntu ( 2.6.29 )
root (hd0,0)
kernel /boot/vmlinuz-2.6.29 ro root=/dev/sda1
initrd /boot/initrd.2.6.29
#
# Entry to chainload into Lilo:
#
title Lilo
rootnoverify (hd0,0)
chainloader +1

```

The first entry is used to boot a Linux kernel. The label of each entry can be set with the *title* command, which will also start a new menu entry. The line *root (hd0,0)* tells Grub to look for any files specified later in this entry inside the first partition of the first hard drive. The *kernel* command specifies a file that will be loaded by grub, once it boots the specific entry. All commands that follow this file will be passed to the Linux kernel as kernel options. Specifically, the *root* kernel option tells the kernel which partition contains the Linux root filesystem.

The second entry with the title *Lilo* is used to chainload into Lilo. The command *rootnoverify* tells Grub the partition where the secondary boot loader is installed. The command *chainloader +1* will read the first block from that partition and pass control to any boot loader that is installed.

After */boot/grub/menu.lst* has been created, Grub must be installed into the MBR with the following command.

```
$ sudo grub-install /dev/sda
```

Note: This has to be done only once! Successive modifications of the Grub configuration file do not make a re-installation necessary, as opposed to Lilo. The next step will be to configure and install Lilo, so that it will chain load back into the MBR where Grub is installed.

### 7.1.2 Lilo

The configuration file for Lilo is */etc/lilo.conf* and looks quite similar to the one Grub uses. It also contains two entries: one for booting a Linux kernel and the other for chain loading Grub.

```

# /etc/lilo.conf
#
boot=/dev/sda1      # install LILO into the first partition of the first hard
                     # drive
timeout=150
root=/dev/sda5       # default root device for 'image' entries
#
# Entry to boot Linux:

```

```

image=/boot/vmlinuz-2.6.25
label=Ubuntu
initrd=/boot/initrd.2.6.25
root=/dev/sda2 # overwrites the global 'root' option
#
# Entry to chainload into Grub:
other=/dev/sda
label=Grub

```

In *lilo.conf*, each entry that does not boot into Linux, starts with *other*, followed by a device name of the partition where the boot loader to be chain loaded is located. In our example, as Grub is installed into the MBR, we specify */dev/sda*, the device file of the first hard drive.

Lilo will be installed with the following command:

```
$ sudo lilo
```

This will install Lilo into the location which was specified by the *root* option in the file *lilo.conf*. Note: Unlike Grub, Lilo has to be re-installed every time you modify its configuration file */etc/lilo.conf*!

## 7.2 Boot Loader Backup/Restore

On Linux, the boot loader can be easily backed up and restored with on-board equipment. It is not necessary to install any Third-party tools to achieve these goals. The command we will use is called *dd*. It is essentially a tool to copy and convert files on a per-byte basis. Unlike the commonly used *cp* it has a few additional arguments that control the amount of bytes copied from one file to the other. The arguments used in this example are explained in the following listing.

```

if=<file> # Copy bytes from <file>
of=<file> # Copy bytes to <file>
bs=N      # Use N bytes per block
count=N   # Copy N blocks

```

In this example *dd* will be used to save, overwrite and restore the boot sector where Lilo is installed. Because Lilo has not been installed into the MBR, it is still possible to boot Linux using Grub in case anything goes wrong in the following procedure. Nevertheless, special care must be taken to type the following commands with the exact values, as any typing error can lead to data loss or an un-bootable system!

The command below will back up the boot sector of the first partition to the file *sda1-bak*.

```
$ dd if=/dev/sda1 of=sda1-bak bs=446 count=1
```

Only the first 446 bytes will be saved because they contain the complete boot sector. The next command will override the Lilo boot sector with zeros and thus make it impossible to chain load to Lilo. This simulates the situation where Lilo is broken due to misconfiguration or hard disk failure.

```
$ dd if=/dev/zero of=/dev/sda1 bs=446 count=1
```

Finally, the next command restores Lilo from the backup created with the first command.

```
$ dd if=sda1-bak of=/dev/sda1 bs=446 count=1
```

### 7.3 Useful commands

- How to know your root partition's device file?

```
$ mount      # prints all mounted partitions  
$ df -h /    # shows free and used disk space for the partition where / is  
               mounted
```

- Showing boot time messages. These are stored in the kernel ring buffer.

```
$ dmesg          # prints the whole buffer  
$ dmesg | grep -i pci # prints information regarding the pci bus
```

# The Linux Init Process

---

## Contents

<b>8.1 Services</b>	29
<b>8.2 System V Init</b>	29
8.2.1 init scripts	30
8.2.2 Runlevel configuration	30
8.2.3 Writing init scripts	31
8.2.4 Default runlevel	32
<b>8.3 Upstart</b>	32
8.3.1 Service control	32
8.3.2 Service configuration files	33
8.3.3 Default runlevel	33

---

This chapter gives an overview about the two Init systems used in Ubuntu, System V Init and Upstart<sup>1</sup>. Both systems manage the set of services that shall be running on a Linux system. They are responsible for monitoring, starting and stopping these services.

## 8.1 Services

Generally, a “service” is a program, which runs in the background and waits for and responds to requests. On Linux, these services typically are called “daemons”. The difference is that a “normal” application or program has an interactive user interface, whereas a service performs tasks in the background, and the user normally is not aware of it.

Two kinds of services can be differentiated: system services and network services. A network service is a program that is running in the background and listening to the network. If a request is coming, it is responding accordingly. Typical network services are Samba, Apache, SSH.

A system service is performing local background tasks on the computer. Typical system services are Cron, Syslogd, X.

## 8.2 System V Init

The System V init system relies on a set of scripts located in the directory `/etc/init.d/` and links to that scripts in the directories `/etc/rcN.d/` where N corresponds to runlevels which can

range from 0 to 6.

### 8.2.1 init scripts

For each installed (network or system) service on the host, there is a script in the folder `/etc/init.d`, which manages proper starting and stopping of the daemons. These scripts can be executed with the parameters “start” and “stop”, for starting or stopping them manually. Most init scripts also accept the parameters “restart” and/or “reload” (for restarting or reloading its configuration), and “status” (for displaying the weather the service is running or not). The following example will stop and restart the ssh daemon:

```
$ /etc/init.d/ssh restart
Restarting OpenBSD Secure Shell server: sshd.
```

### 8.2.2 Runlevel configuration

A *set of init scripts*, which define, which services shall be started, and which not, is called a “runlevel”. This will define the state, in which the computer will be. Under most Linux distributions, there are seven runlevels (0 to 6), which are organized in respective folders. Under Ubuntu, runlevel 2 is the default runlevel and contains all necessary services for a normal desktop operation. This means, on boot up, the computer is entering runlevel 2 and thus starting all relevant configured services. Runlevel 1 starts the system in single user mode. This will only allow root to login and does not provide a graphical login. Runlevel 0 is reserved for shutdown, runlevel 6 for reboot. The other runlevels can be configured and customized to specific user needs.

For each runlevel, there exists a folder in “`/etc`”. Under Ubuntu, they have the names `/etc/rc0.d`, `/etc/rc1.d`, `/etc/rc2.d`, `/etc/rc3.d`, `/etc/rc4.d`, `/etc/rc5.d`, `/etc/rc6.d`. These folders contain *symbolic links* to the init scripts in “`/etc/init.d`”.

The names of the links are starting with either “S” or “K”. When a runlevel is entered (switched to), all links starting with a “S” are executed with the parameter “start” and all links starting with a “K” are executed with the parameter “stop”. That way, it can be easily configured, which services to start and which to kill in a runlevel.

After the “S” or the “K” comes a two-digit number in the filename. When entering a runlevel, the scripts containing the respective folder will be executed exactly in the order of these numbers. I.e. lower numbers are executed before higher numbers. That way, the order of the starting or stopping of the daemons can be controlled.

Runlevels can be switched manually with the “init” command; the “runlevel” command displays the current and the previous runlevel.

### 8.2.3 Writing init scripts

In principle, init scripts can be written in any programming language. As a matter of fact, almost all init scripts are shell scripts. The best practice is to copy an existing script from the distribution and adjust it in the proper way. The LSB ( Linux Standard Base ) defines a set of standard headers which must be present in an LSB-conforming init script. These headers are shown in the following code block. Tools like *chkconfig* need them to automatically create the appropriate links in the corresponding runlevel directories.

```
### BEGIN INIT INFO
# Provides:           service name
# Required-Start:    services that must be started before
# Required-Stop:     services that must be stopped before
# Should-Start:      services that should be started before
# Should-Stop:       services that should be stopped before
# Default-Start:     runlevels where service should be started (e.g. 2 3 4 5 )
# Default-Stop:      runlevels where service should be stopped (e.g. 0 1 6 )
# Short-Description: short description of service
# Description:       long description of service
### END INIT INFO
```

The next codeblock shows a typical body of a System V init script. It contains a case statement which handles the typical arguments an init script can be called with (start/stop/restart/status).

```
case "$1" in
  start)
    # start service
    ;;
  stop)
    # stop service
    ;;
  restart)
    # restart service
    ;;
  status)
    # print status of service ;;
  *)
    echo "Usage: /etc/init.d/service {start|stop|restart|status}"
    exit 1
esac

exit 0
```

Most of the init scripts include the file */lib/lsb/init-functions* which contains methods for recurrent tasks like starting/stopping a service or printing status information to the screen. Some of the methods are listed below.

**log\_begin\_msg "message":** Give output on begin of an action

**log\_end\_msg N:** Output the result of an action, where *N* specifies the return code

**start-stop-daemon [options] command:** Start a service in the background

More information about LSB compliant init scripts can be found here: [http://refspecs.freestandards.org/LSB\\_3.1.0/LSB-Core-generic/LSB-Core-generic/tocsysinit.html](http://refspecs.freestandards.org/LSB_3.1.0/LSB-Core-generic/LSB-Core-generic/tocsysinit.html)

### 8.2.4 Default runlevel

The runlevel the init system will enter when the system has been powered on is called the default runlevel. With System V init, the default runlevel can be changed in two different ways. The first one, which takes precedence over the second one, is specifying the default runlevel as a parameter to the kernel command line. For this method to work, a boot loader like grub must be used which allows to give arguments to the kernel at the boot prompt. The second possibility is to add a certain line to the file `/etc/inittab`. In the following code block, the number behind the first double colon specifies the default runlevel.

```
id:1:initdefault:
```

## 8.3 Upstart

Upstart is a replacement for the traditional System V Init system. It is designed to be more flexible than the previous system. This includes the possibility to start and stop jobs asynchronously instead of waiting for each job to finish. Starting and stopping of jobs is not limited to runlevel changes, but is triggered by “events” that are emitted whenever something in the system changes, e.g. a file system is mounted or a usb device is plugged in.

The following table summarizes some of the differences between Upstart and System V init.

	Directory for job definitions	Start/stop triggered by	Start/stop configuration done by
SysV Init	<code>/etc/init.d</code>	Runlevel change	Links in <code>/etc/rcN.d</code>
Upstart	<code>/etc/init</code>	events	Conditions contained in job definitions

### 8.3.1 Service control

Upstart provides the command `initctl` for controlling and monitoring the state of system services. Calling `initctl` with the arguments `start/stop/restart/status <jobname>` will start, stop, restart and respectively show the status of an Upstart service. The command `initctl list` will list all services known to Upstart together with their current status. With the command `initctl emit <event>`, a custom named event can be triggered. For a complete listing of all available options to the `initctl` command, please see the associated man page.

### 8.3.2 Service configuration files

Upstart aims to be compatible to System V init, that means, old System V init scripts will continue to work with Upstart. These are executed by a special Upstart job which is triggered whenever the system changes its runlevel. Of course, these services will not benefit from the new features of Upstart, like parallel execution and event based starting/stopping. To support these features, Upstart introduces its own style of service configuration files. An example job definition for the service 'whologger' is shown in the next code block.

```
#  
# file /etc/init/whologger.conf  
#  
start on runlevel [2345]  
stop on runlevel [!2345]  
  
expect fork  
  
exec /usr/sbin/whologger
```

This job definition causes the example service */etc/sbin/whologger* to be started in the background whenever the system enters one of the runlevels 2,3,4 or 5. In case the system enters any other runlevel, the service will be automatically stopped.

Consecutively, some of the possible keywords an Upstart job configuration file may contain are summarized.

**start on <events>:** lists events that start the job

**stop on <events>:** lists the events that stop the job

**script / end script:** encloses shell script to run

**exec <command>:** Starts one command

**expect fork:** The service is expected to fork into background and will be supervised

An <event> can be one of the following:

**startup:** emitted when the system starts up

**started/stopped <job>:** emitted when a job is started/stopped

**runlevel [n]:** emitted when the system enters runlevel n

### 8.3.3 Default runlevel

Additionally to the methods for changing the default runlevel on a system V init configuration, the default runlevel for Upstart can be altered in the file */etc/init/rc-sysinit.conf*. Search for the following line and change it accordingly.

```
env DEFAULT_RUNLEVEL=2
```

Further information about Upstart can be found at <http://upstart.ubuntu.com/wiki/>.

# The Linux Init Process – Exercises

---

## Contents

<b>9.1 Upstart exercise – Network Monitor . . . . .</b>	<b>35</b>
9.1.1 Task description . . . . .	35
9.1.2 Example output . . . . .	36
9.1.3 Example solution . . . . .	36

---

In this chapter an exercise is given which combines two tasks:

1. Writing a service that accomplishes the simple task of checking whether a given host is reachable.
2. Writing an Upstart service configuration file for that service.

The first part contains the task description and the second part presents a proposed example solution.

## 9.1 Upstart exercise – Network Monitor

### 9.1.1 Task description

1. Write a service “NetworkMonitor” according to the following definition
  - The service should be a bash script in the file `/usr/sbin/NetworkMonitor`
  - The service should have the following synopsis :

```
$ NetworkMonitor seconds
```
  - The service should process the following steps repeatedly in an interval given by the `seconds` argument
    - (a) Read a list of hosts from the file `/etc/NetworkMonitor.hosts`
    - (b) Test the reachability of each host ( e.g. by using ping )
    - (c) Log the result of each test to the file `/var/log/NetworkMonitor.log`
2. Write an Upstart job for the service “NetworkMonitor”
  - Write the job into the file `/etc/init/networkmonitor.conf`
  - The job should be started when networking is available
  - The Job should stop whenever networking is disabled

### 9.1.2 Example output

Content of /var/log/NetworkMonitor.log:

```
May 31 10:56:42: Testing hosts from file '/etc/NetworkMonitor.hosts'
May 31 10:56:44: Host 192.168.0.1 is up
May 31 10:56:46: Host 192.168.0.2 is down
May 31 10:56:48: Host 192.168.0.3 is down
May 31 10:56:50: Host 192.168.0.4 is up
May 31 10:56:52: Finished testing 4 hosts
```

Content of /etc/NetworkMonitor.hosts:

```
192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.4
```

### 9.1.3 Example solution

The file /usr/sbin/NetworkMonitor:

```
#!/bin/bash

hostfile=/etc/NetworkMonitor.hosts
logfile=/var/log/NetworkMonitor.log

print_to_log () {
    echo "date:$1" >> $logfile
}

test_hosts () {
    print_to_log "Testing hosts from file '$hostfile'"
    cat $hostfile | while read line ; do
        ping $line -c 1 >/dev/null 2>&1 &&
            print_to_log "Host $line is up" ||
            print_to_log "Host $line is down" ;
    done
    print_to_log "Finished testing `wc -l $hostfile | cut -d' ' -f 1` hosts"
}

if [ -z "$1" ] ; then
    echo "usage: basename $0"
    exit 1
fi

seconds=$1

while true; do
```

```
test_hosts
sleep $seconds
done &
```

The file `/etc/init/networkmonitor.conf`:

```
# /etc/init/networkmonitor.conf
# =====
start on ( started networking or
            started network-manager or
            manual-monitor-start )

stop on ( stopped networking or
            stopped network-manager or
            manual-monitor-stop )

pre-start script
    echo -n "NetworkMonitor started:" >> /var/log/NetworkMonitor.log
    date >> /var/log/NetworkMonitor.log
end script

post-stop script
    echo -n "NetworkMonitor stopped:" >> /var/log/NetworkMonitor.log
    date >> /var/log/NetworkMonitor.log
end script

expect fork
exec /usr/sbin/NetworkMonitor 5
```

In this solution, the “NetworkMonitor” service will be started and stopped together with the jobs “networking” and “network-manager”. Additional it will be started when Upstart receives the event “manual-monitor-start” and stopped when it receives the event “manual-monitor-stop”. The example also shows how to use the Upstart keywords *pre-start* and *post-stop* to trigger actions before the service is started respectively after it has been stopped.



# The Linux Desktop

---

## Contents

<b>10.1 General overview . . . . .</b>	<b>39</b>
<b>10.2 The X Server . . . . .</b>	<b>39</b>
10.2.1 Special commands . . . . .	40
<b>10.3 Display Manager . . . . .</b>	<b>40</b>
<b>10.4 Window Manager . . . . .</b>	<b>41</b>
<b>10.5 Desktop Environment . . . . .</b>	<b>41</b>

---

This chapter gives an overview about the different parts that form the Linux graphical user interface. The Linux desktop is typically the first thing novel users face when they start using Linux. The way the graphic user interface is designed on Linux differs fundamentally from the implementation on other operating systems. Therefore it is eligible for a new user to have a general overview about the software systems that are involved and the way they interact with each other. This chapter will only explain the user land part of the Linux graphic stack. Graphic drivers, which are part of the Linux kernel will not be covered.

## 10.1 General overview

Figure 10.1 shows an overview about the different software involved in the Linux desktop. The way they are ordered from the user on the top to the hardware on the bottom, should roughly reflect usage dependency between the software. Software on the top uses services of the software located below them. The next sections will give a short introduction to each of them.

## 10.2 The X Server

The X Window System<sup>1</sup> (commonly X or X11) is a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers. It creates a hardware abstraction layer where software is written to use a generalized set of commands, allowing for device independence and reuse of programs on any computer that implements X.[wik 2010e]

---

<sup>1</sup><http://www.x.org/wiki/>

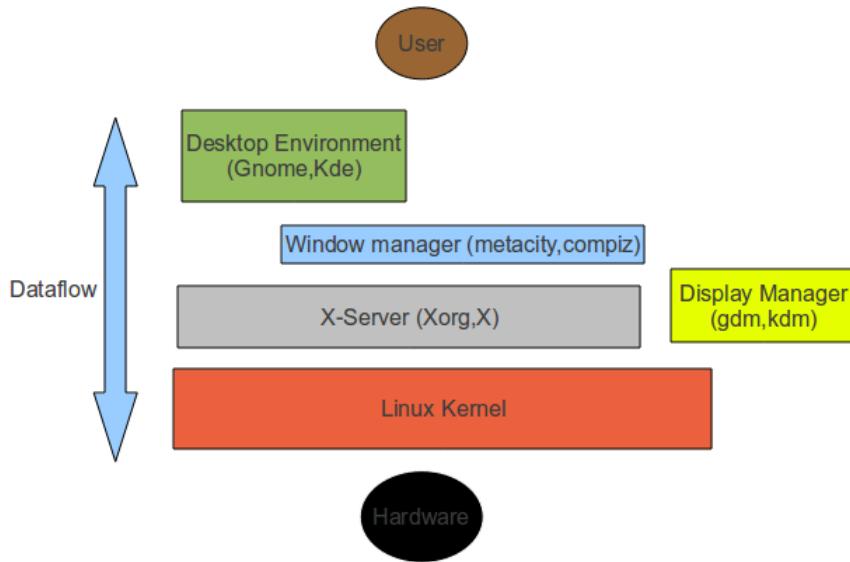


Figure 10.1: Linux graphic stack overview.

### 10.2.1 Special commands

**xwininfo:** Shows informations about windows related to the X Server

**xkill:** Provides a graphical way to close the window resource of an

**startx:** Starts an Xserver and executes all commands in the file `/home/<user>/ .xinitrc`  
The X Server terminates when the last command in the list exits.

The following listing shows an example for the file `/home/<user>/ .xinitrc`. First, the window manager `metacity` will be started and afterwards a terminal emulation program called `xterm`. In this example, the X Server will terminate when the user closes the terminal window.

```
# /home/user/.xinitrc
metacity &
xterm
```

## 10.3 Display Manager

In the X Window System, an X display manager runs as a program that allows the starting of a session on an X server from the same or another computer. A display manager presents the user with a login screen which prompts for a username and password. A session starts when the user successfully enters a valid combination of username and password. [wik 2010d] Typical display managers are `gdm` for the Gnome desktop environment, `kdm` for the Kde desktop environment and `xdm` which is part of the X software system. On Ubuntu, both `gdm` and `kdm` are services that can be started and stopped with the following commands:

```
/etc/init.d/gdm start/stop  
/etc/init.d/kdm start/stop
```

## 10.4 Window Manager

A window manager is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface.[1] Most window managers are designed to help provide a desktop environment. They work in conjunction with the underlying graphical system which provides required functionality such as support for graphics hardware, pointing devices, and a keyboard, and are often written and created using a widget toolkit.

The elements usually associated with window managers are those which allow the user to open, close, minimize, maximize, move, resize, and keep track of running windows, including window decorators. [wik 2010c]

Examples of window managers are *metacity*, *kwin* and *compiz* the latter being mostly used to provide graphic effects like transparent windows and animations.

## 10.5 Desktop Environment

In graphical computing, a desktop environment (DE) commonly refers to a style of graphical user interface (GUI) derived from the desktop metaphor that we see on most modern personal computers. These GUIs help the user in easily accessing, configuring, and modifying many important and frequently accessed specific operating system (OS) features.

A desktop environment typically consists of icons, windows, toolbars, folders, wallpapers, shortcuts and desktop widgets. [wik 2010a]

Prominent desktop environments are *Kde*<sup>2</sup> and *Gnome*<sup>3</sup>. On an Ubuntu system, these can be installed by the following commands.

```
sudo aptitude install kubuntu-desktop # installs the kde desktop  
sudo aptitude install ubuntu-desktop # install the gnome desktop
```

---

<sup>2</sup><http://www.kde.org/>

<sup>3</sup><http://www.gnome.org/>



# Shell scripting

---

## Contents

<b>11.1 Overview</b>	43
<b>11.2 Scripts</b>	43
<b>11.3 Variables</b>	44
<b>11.4 help</b>	45
<b>11.5 Starting Scripts</b>	45
<b>11.6 “.” and “..”</b>	45
<b>11.7 script arguments</b>	46
11.7.1 if and test	47
<b>11.8 Redirections</b>	49
11.8.1 Input & Output Redirection	49
11.8.2 The Pipe	50
<b>11.9 Command Substitution</b>	51
<b>11.10 Loops</b>	51
11.10.1 for loop	52
<b>11.11 while loop</b>	53

---

A shell is an application that reads and executes commands. It has some build-in commands (like “cd”, “ls” and “echo”), and can execute other programs.

## 11.1 Overview

A shell is a textual user interface (UI). It can start, direct and stop programs like a graphical user interface (GUI). In a textual UI the user has to type commands, which will be read and interpreted by a command-line interpreter.

## 11.2 Scripts

Modern shells can work in two modes: the “interactive mode” is used for starting programs, moving files and so on. Commands are read from stdin (the keyboard in a terminal window).

The “non-interactive mode” is used, whenever a file (the “shell script”) is used to send commands to the shell. The “bash” (Bourne-Again SHell) works the same way in both modes. That can

be different with other shells. The shell simply reads the contents of the file (line by line) and executes it - as if the commands were given via keyboard.

### 11.3 Variables

Not build-in programs are searched for in a list of filesystem paths. This list is saved in an environment variable called PATH.

The contents of shell variables can be accessed by “dereferencing” them. That can be done by prepending a \$ sign to the variables name. They can then be printed with echo:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/opt/bin:/usr/x86_64-pc-linux-gnu/gcc-bin/4.3.3
```

Other useful (build-in) variables are:

```
$ echo $SHELL
/bin/bash
$ echo $HOME
/home/daniel
$ echo $USER
daniel
$ echo $PWD
/home/daniel/ziik/svn
```

You can define your own variables by “assigning” a value to them:

```
$ MY_VAR="hello world"
$ echo $MY_VAR
hello world
```

To append text to a variable:

```
$ MY_VAR="$MY_VAR and universe"
$ echo $MY_VAR
hello world and universe
```

Before an assignment (`variable=value`) happens, the right side is “evaluated” (all computation is done). In the first line `$MY_VAR` got replaced by its value (we say it was “dereferenced”), and a new string was created. That new string (“hello world and universe”) was then assigned to the variable `$MY_VAR`, overwriting the old value.

## 11.4 help

The man page for bash is very long and complicated. For most simple purposes the build-in “help” function is enough and fast to use. Try these:

```
$ help cd  
$ help for  
$ help test
```

## 11.5 Starting Scripts

A script can be executed by “feeding” the file containing the script to the interpreter (the shell):

```
$ bash myscript  
do something
```

It is also possible to write the interpreter to use into the script file in its first line:

```
#!/bin/bash  
  
echo "do something"
```

Then the script can be executed directly, and the interpreter (the shell) will be used automatically:

```
$ chmod a+x myscript  
$ ./myscript  
do something
```

## 11.6 “.” and “..”

The scripts name has to be prepended with “`./`”, because the correct working directory (CWD) is not in the shell’s search path. That is a security measure. In UNIX, the “`.`” (dot) is the CWD. So to execute any program in the CWD, you have to run “*this directory/program*”, which translates into “`./program`”.

To see what I mean, run

```
$ ls
```

and

```
$ ls .
```

It shows the same.

In fact, in *every* directory in a UNIX system (even “empty ones”) you will find two directories: “.” and “..”. Where “.” points to the directory itself, and “..” to the parent of this directory<sup>1</sup>.

Running

```
$ ls /tmp
```

and

```
$ ls /tmp/.
```

is the same. Even

```
$ ls /tmp/././.
```

is just “/tmp”.

## 11.7 script arguments

When a program is started, the complete command line that was used to start it is made known to it. This way programs receive options.

In the shell the special variables “\$0”, “\$1”, “\$2” and so on are used for this purpose. “\$0” is the name of the program that was used to start it (normally that is the name of your script), “\$1” is the first argument passed to it, “\$2” is the second argument, and so.

You can use them like this:

---

<sup>1</sup>Except for the root directory, where both point to the same directory.

```
#!/bin/bash

echo "my programs name: $0"
echo "first argument : $1"
echo "second argument: $2"
echo "third argument: $3"
```

```
$ chmod a+x testargs
$ ./testargs apple pear
my programs name: ./testargs
first argument : apple
second argument: pear
third argument :
```

In bash it is OK to use variables that have not been assigned a value (like “\$3” here). In other programming languages that is often an error.

### 11.7.1 if and test

If you would like to augment your script to test for unused arguments, you can use the “test” command (see “`$ help test`”):

```
#!/bin/bash

echo "my programs name: $0"

if [ ! -z $1 ]; then
    echo "first argument : $1";
fi

if [ ! -z $2 ]; then
    echo "second argument: $2";
fi

if [ ! -z $3 ]; then
    echo "third argument : $3";
fi
```

```
$ ./testargs apple pear
my programs name: ./testargs
first argument : apple
second argument: pear
```

The “**if**” build-in command does only execute the “**then**” part, if its test was “true”. Its simple syntax is:

```
if <test>; then <command>; fi
```

Its complete syntax includes “**else**” and “**elif**” parts. See “`$ help if`”:

```
if COMMANDS; then COMMANDS; [ elif COMMANDS; then COMMANDS; ]... [ else COMMANDS; ] fi
```

“`<test>`” can be any command or *build-in test*. Such a build-in test is done between `[ ]`. “`!`” means “*not*”, inverting the result of the following test. “`-z`” checks if the following string is empty. So the whole test (“`[ ! -z $1 ]`”) returns “true” if “`$1`” is not empty. Only in this case, the “**then**” part is executed. It is important to write spaces around the first square bracket.

To read about all possible build-in test, run “`$ help test`”.

If you wish to note every copy operation you do, you could write a small script that does that:

```
#!/bin/bash

if [ -z "$1" -o -z "$2" ]; then
    echo "You must supply two arguments."
    exit 1
fi

echo "$(date) Copying $1 to $2" | tee -a /tmp/copy.log

cp -v $1 $2
```

Example:

```
$ chmod a+x mycopy
$ ./mycopy /etc/fstab
You must supply two arguments.
$ ./mycopy /etc/fstab /tmp/test.txt
Mon Aug 3 15:03:09 CEST 2009 Copying /etc/fstab to /tmp/test.txt
'/etc/fstab' -> '/tmp/test.txt'
```

Now check the contents of the log file (“`/tmp/copy.log`”). Copy another file, and check the log file again.

To understand how “`$(date)`” works, read further down (Section 11.9 on page 51). The important thing now is, that the output of the “**date**” program appears in its place.

The **pipe** (“`|`”) is also explained further down (Section 11.8.2 on page 50). In short: the output of the left command is used as input to the right command (in this case “**tee**”).

The “`tee`” program takes input from another command, and copies this input to one or more files, and writes it also back to the terminal. The “`-a`” option makes it **append** to the file, instead of truncating it, before writing to it.

Example:

```
$ echo "test\ntest" | tee logfile
test test
$ cat logfile
test test
$ echo "more\ntesting" | tee -a logfile
more testing
$ cat logfile
test test
more testing
```

## 11.8 Redirections

### 11.8.1 Input & Output Redirection

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell.

But before we dive into redirection, a small excursion into UNIX file connectors (so called “file descriptors” for programs) is necessary:

- “Standard input” (`stdin`) is by default connected to the keyboard input from the user in a terminal.
- “Standard output” (`stdout`) is by default connected to the output to a terminal. This is for the normal output a program writes to you.
- “Standard error” (`stderr`) is by default connected to the output to a terminal. This is for error messages from programs.

Normally you don’t have to worry about these things. Your keyboard will always be connected to the “`stdin`” of the program you are using. “`stdout`” and “`stderr`” will both be connected to the terminal from which you have started the program. This way you can read both, normal output and error messages, in the same terminal.

When you want to *redirect* input or output of a program though, it is important to redirect the right thing :-)

If you wish to save the output of a command to a file, you use the right arrow (“`>`”):

```
$ ls / > /tmp/myroot.txt
```

This redirects the output of “ls” into the file “/tmp/myroot.txt” (instead of showing it on your terminal). “stdout” has been redirected. If an error would have occurred, it would show, because “stderr” has not been redirected, and is still connected to your terminal:

```
$ ls /doesnotexist > /tmp/myroot.txt
ls:  cannot access /doesnotexist:  No such file or directory
```

If you wish to *append* the output of a command to a file, you must use “»”:

```
$ ls / > /tmp/myroot.txt
$ ls / » /tmp/myroot.txt
$ ls / » /tmp/myroot.txt
```

Now you have written three times the content of “/” in the file. (The first time “>” was used, to truncate the file, in case there was already something inside.)

To redirect “stderr”, use “2>”:

```
$ ls /doesnotexist 2> /tmp/myroot_err.txt
```

This time “stderr” was redirected. The error message is now in “/tmp/myroot\_err.txt”.

To redirect both use:

```
$ ls /etc /doesnotexist > /tmp/myroot_stdout.txt 2> /tmp/myroot_stderr.txt
```

This will send the listing of “/etc” to “/tmp/myroot\_stdout.txt” and the error message to “/tmp/myroot\_stderr.txt”.

### 11.8.2 The Pipe

The pipe (“|”) is another way to redirect input and output of programs. It is a very powerful tool, that brings interprocess communication to your fingertips. Whenever you want the output of one program to be processed by another program, you can connect the “stdout” of the first to the “stdin” of the second program with the pipe.

```
$ cat /var/log/syslog | tail
[ ...  ommitting output...  ]
```

`cat` displays the contents of the `/var/log/syslog` on the screen, but the output has been connected to the input of the program `tail`. So only the last ten lines of it are shown.

```
$ sudo find /home -type f | wc -l  
78501
```

The last example counts, by the use of the command `wc`, all files printed by `find` in all users home directories. Please note, that only the command directly following the “`sudo`” is run with root privileges, the command after the pipe is running with normal permissions.

## 11.9 Command Substitution

Command substitution allows the output of a command to replace the command name. There are two forms: `$(command)` and ‘`command``.

Bash performs the expansion by executing `command` and replacing the command substitution with the standard output of the command, with any trailing newlines deleted.

### Examples:

This will print the message “The current directory is” and then the *output of the command “`pwd`”*:

```
echo "The current directory is: $(pwd)"
```

The following script will print “Good morning”, if the time is before 12:00 PM, and else “Good afternoon”:

```
#!/bin/bash  
if [ $(date +%k) -lt 12 ]      # "$(date +\%k)" is replaced with the actual hour  
then echo "good\u00famorning"  
else echo "good\u00f7afternoon"  
fi
```

## 11.10 Loops

A loop is a bash programming language statement which allows code to be repeatedly executed. A loop is classified as an iteration statement (the loop specifications). This defines under which conditions, when, how often etc. the loop is executed.

Several types of loops exist in bash. Two of these, `for` loops and `while` loops shall be discussed here:

### 11.10.1 for loop

Typically, a for loop consists of the following statements:

```
for <variable> in <range>
do
    list of commands to loop
done
```

A **for** loop is a *counting* loop, which, while it executes the loop, cycles a specified variable through a sequence of items. Items can be numbers, alphanumeric characters, or a list of words. The loop (the commands between “do” and “done”) is executed as many times as there are items in the sequence. During each cycle of the loop, the used variable has the value of the particular sequence count.

### Examples

The following script will print the message “Welcome x times” five times and in each line print the actual number of the loop count as “x”:

```
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

Ranges can also be defined with the **seq** command. The syntax of this command is (example):

```
seq 5 2 15
```

This will return a sequence of numbers between 5 and 15, with a jump of 2 (5,7,9,11,13,15). With this, the for loop in the example above could be changed as follows:

```
#!/bin/bash
for i in $(seq 1 1 5)
do
    echo "Welcome $i times"
done
```

The contents of a folder can be easily used as a sequence, too. The following script will go through all files in the folder **/etc** and print its type:

```
#!/bin/bash
for f in /etc/*
do
    file $f
done
```

The following script will go through all files in the current directory that are ending with .sh and change their permissions to 755:

```
#!/bin/bash
for f in *.sh
do
    chmod 755 $f
done
```

The following script will go through all files in the current folder and ask to delete them:

```
#!/bin/bash
for f in *
do
    echo -n "File \"$f\" is "
    file -b $f
    echo -n "Delete it? (yes/no) "

    read d

    if [ $d == yes ]
        then rm $f
        echo "File \"$f\" deleted!"
        echo
    fi

done
```

## 11.11 while loop

Typically, a `while` loop consists of the following statements:

```
while [<expression> is true]
do
    <list of commands to loop>
done
```

A **while** loop is executing a list of commands as long as an expression is true. This means, it will test if the used expression (test) is the case, and if yes, it will execute the loop. This makes it possible to infinitely execute a loop, until a certain event is happening.

The iteration statement of the **while** loop is similar to a **if** statement.

## Examples

The following script likes it, when you feel good. The loop consists of a **read** command. It will be executed, as long as the variable is something different than “good”. The exclamation mark *negates* the test; i.e. it is true, if the variable is *not* equal “good”.

```
#!/bin/bash
while [ "$i" != good ]
do
    echo "how do you feel?"
    read i
done
echo "that's great! :)"
```

The following script prints the number of files from a folder the user can pick from a given list:

```
#!/bin/bash
choice=4
while [ $choice == 4 ]
do
    echo "Count number of files in:"
    echo "(1) /etc"
    echo "(2) /var/log"
    echo "(3) /lib"
    read choice
    if [ $choice == 1 ]
    then echo "/etc has `ls -l /etc| wc -l` files."
    else
        if [ $choice == 2 ]
        then echo "/var/log has `ls -l /var/log|wc -l` files."
        else
            if [ $choice == 3 ]
            then echo "/lib has `ls -l /lib|wc -l` files."
            else
                echo "Please enter 1, 2 or 3!"
                choice=4
            fi
        fi
    fi
done
```

This is also an example for “nested if statements”. This means, part of the **else** section of one

`if` statement is *another if* statement, making it possible to issue a further “`if`” test, if the first test was false. Part of the `else` section of the second `if` statement is a third `if` statement.

At first, a “testing” variable (`choice`) is introduced with the value `4` in this script and the condition for the repetition of the loop is, if the variable `choice` has the value `4`. The `read` command stores the input from the keyboard into the same variable. Only if all three `if` tests were false, the user will be prompted with a note (`Please enter 1, 2 or 3!`) and the variable is set to `4` again, so that the loop will be repeated, in case the user did not enter a meaningful input.



# Shell scripting - exercises

---

## Contents

<b>12.1 Overview</b>	57
<b>12.2 First Exercise: Directory Watch (***)</b>	58
<b>12.3 Second Exercise: File system checker (***)</b>	58
<b>12.4 Third Exercise: File Size Analyzer (**)</b>	59
<b>12.5 Fourth Exercise: File Checker (**)</b>	61
<b>12.6 Fifth Exercise: Batch Rename (*)</b>	62
<b>12.7 Sixth Exercise: Font Finder (*)</b>	63
<b>12.8 Seventh Exercise: Size Sorter (**)</b>	64
<b>12.9 Eighth Exercise: File Finder (**)</b>	65
<b>12.10 Ninth Exercise: Find Web Server (***)</b>	66
<b>12.11 Tenth Exercise: Archiver (**)</b>	67
<b>12.12 Eleventh Exercise: Create svn home directory (*)</b>	68
<b>12.13 Twelfth Exercise: Arping all 254 hosts in a given Class C network (*)</b>	68
<b>12.14 Thirteenth Exercise: Create a list of files that are opened by a process (***)</b>	69

---

Besides knowing the basics of shell scripting it is very necessary to just get practice. The best way to learn and to become familiar with shell scripting is to try to solve particular task, doing research on the needed tools (commands), adapt existing solutions and finally create own ones. The following exercises are given to give a broad overview about possible tasks and maybe impulses for creating new ones. Also exemplary solutions are given, but of course there are many ways to solve these tasks.

## 12.1 Overview

In the following thirteen different task will be introduced. Each task includes a brief description, a definition of the syntax (how the script should be called from the command line) and a hint of which CLI commands can be used to solve each exercise. Finally there will be one solution for each of these examples. Additional the difficulty of each example is indicated from one star (\*) till three stars (\*\*\*)�.

## 12.2 First Exercise: Directory Watch (\*\*\*)

**Commands used:** logger, du

**Synopsis:** watchdir <directory> <maximum size in bytes>

**Description:** Write a script that periodically checks if the size of a given directory exceeds a given maximum. The script should log the following events to the syslog daemon:

- The current size is only 5% less then the maximum size.
- The current size has exceeded the maximum size
- The current size has fallen below the maximum size

**Possible solution:** watchdir.sh

```
#!/bin/bash

if [ -z "$1" -o -z "$2" ] ; then
    echo "usage: watchdir dir size"
    exit 1
fi

state=0

while true ; do

let "size='du -sb $1 | cut -f1'"

if let "$size >= $2" && test "$state" != "1" ; then
    logger "Size of $1 has exceeded $2"
    state=1
elif let "$size < $2" && let "$size >= $2/100*95" && test "$state" != "2" ; then
    logger "Size of $1 is almost $2"
    state=2
elif let "$size < $2/100*95" && test "$state" != "3" ; then
    logger "Size of $1 has fallen below $2"
    state=3
fi

sleep 5

done
```

## 12.3 Second Exercise: File system checker (\*\*\*)

**Commands used:** fsck, mount, fdisk

**Synopsis:** filesystemcheck [show|check] [device file]

**Description:** Write a script that lets the user choose to check a given file system. When the user calls the script with the argument 'show', the script should output all existing partition devices on the first two hard drives. If the script is called with the argument 'check', an additional device file must be given.

The scripts should then perform a file system check on this device. The script should output an appropriate message if the check was successful or not. It should also not check a device that is already mounted or does not exist.

**Possible solution:** filesyscheck.sh

```
#!/bin/bash

usage="usage: filesystemcheck [show|check] [device file]"

if [ -z "$1" ] ; then
    echo $usage
    exit 1
fi

if [ "$1" == "show" ] ; then
    ls /dev/sda* 2>/dev/null
    ls /dev/sdb* 2>/dev/null

elif [ "$1" == "check" ] ; then
    if [ ! -e "$2" ] ; then
        echo $usage
        exit 1
    fi

    mount | grep -q $2 && { echo "$2 is mounted, not checking." ; exit 1 ; }
    cat /proc/swaps | grep -q $2 && { echo "$2 is swap space, not checking." ; exit 1 ; }

    echo "Checking $2..."

    sudo fsck $2 > /dev/null 2>&1 && echo "Filesystem check on $2 successful" ||
        echo "Filesystem check on $2 failed."
else
    echo $usage
fi
```

## 12.4 Third Exercise: File Size Analyzer (\*\*)

**Commands used:** find, du

**Synopsis:** filesizeanalyze [directory] [-n|-u] [name|user]

**Description:** Write a script that calculates the total size of files inside a directory that match certain criteria. The script should search recursively for any file inside the directory given as an argument and print out the cumulated size of these files. If the script is called with the '-n'

option, it should look for files containing 'name'. If it is called with the '-u' option, it should look for files owned by 'user'.

If no directory is given, it should search in the current directory. Proper error handling should be added, e.g. if the user does not exist etc.

**Possible solution:** filesizeanalyze.sh

```
#!/bin/sh
#Script generated by ScriptBuilder v0.1

PARAMS=$#
ACTUALDIR='pwd'
RESULT="No Files Found, Sorry"

if [ "$PARAMS" = "0" ]; then
echo "> Info: Analysing current dir with no options"
FILES='find $ACTUALDIR'

elif [ "$PARAMS" = "1" ]; then
echo "> Info: Analysing Directory $1"

if [ ! -d $1 ]; then
echo "$1 is not a directory"
exit 1
fi
ACTUALDIR=$1
FILES='find $ACTUALDIR'

elif [ "$PARAMS" = "2" ]; then
echo "> Info: Analysing $ACTUALDIR with OPTION $1 and PARAMENTER $2"

if [ "$1" != "-u" -a "$1" != "-n" ]; then
echo "$1 is not a valid option"
exit 1
fi

if [ "$1" = "-u" ]; then
USER=$2
id $USER > /dev/null 2>&1
if [ "$?" -ne "0" ]; then
echo "User $USER is not existing"
exit 1
fi
FILES='find $ACTUALDIR -user $USER -type f'
fi

if [ "$1" = "-n" ]; then
NAME=$2
FILES='find ${ACTUALDIR} -iname "*${NAME}*" -type f'
fi

elif [ "$PARAMS" = "3" ]; then
echo "> Info: Analysing $1 with OPTION $2 and PARAMENTER $3"

if [ ! -d $1 ]; then
echo "$1 is not a directory"
exit 1
fi
```

```

        fi

if [ "$2" != "-u" -a "$2" != "-n" ]; then
    echo "$2 is not a valid option"
    exit 1
fi
ACTUALDIR=$1

if [ "$2" = "-u" ]; then
    USER=$3
    id $USER > /dev/null 2>&1
    if [ "$?" -ne "0" ]; then
        echo "User $USER is not existing"
        exit 1
    fi
    FILES='find $ACTUALDIR -user $USER -type f'
fi

if [ "$2" = "-n" ]; then
    NAME=$3
    FILES='find ${ACTUALDIR} -iname "*${NAME}*" -type f'
fi
fi

echo "> ====="

if [ ! -z "$FILES" ]; then
RESULT='du -hc --apparent-size $FILES 2>/dev/null | grep -i "total" | cut -f 1'
echo "> Files: "
echo "$FILES"
fi

echo "> ====="
echo "> Total size of found files in directory *** $ACTUALDIR *** is: *** $RESULT ***"
echo "> ====="

```

## 12.5 Fourth Exercise: File Checker (\*\*)

**Commands used:** find, md5sum

**Synopsis:** filechecker option dir1 dir2

**Description:** Write a script to fulfill the following described tasks:

- The script should accept 3 command line parameter
- The script should read the content of directory1 recursively
- The script should calculate the md5checksum for each file
- The script should write all processed checksums into one file in the given directory2
- You have to ensure that the directory2 is unique and valid and that no existing data will be overwritten, so you have to rename the dir2 parameter if that directory already exists

(for example append the actual date to the directory name).

- For options the script should interpret -o (for open file inside an editor after writing) and -w for only write to file. All unsupported options should lead to an suitable message.

**Possible solution:** `filechecker.sh`

```
#!/bin/bash

usage="usage: fileschecker -o|-w dir1 dir2"

if [ -z "$1" -o -z "$2" -o -z "$3" \
-o "$1" != "-o" -a "$1" != "-w" \
-o ! -d "$2" ] ; then
echo $usage
exit 1
fi

test -d "$3" && dir="$3`date`" || dir="$3"
mkdir "$dir"

find $2 -type f -print0 | xargs -0 md5sum > "$dir"/md5sums
```

## 12.6 Fifth Exercise: Batch Rename (\*)

**Commands used:** mv

**Synopsis:** batchrename [directory] ext1 ext2

**Description:** Write a script that renames all files with the extension 'ext1' to contain the extension 'ext2'. The files should be searched in the given directory. If no directory is given, it should search in the current directory. Proper error handling should be done.

**Possible solution:** `batchrename.sh`

```
#!/bin/bash

usage="usage: batchrename [dir] ext1 ext2"

let $# < 3" && { dir=.;ext1=$1;ext2=$2; } ||
{ dir=$1;ext1=$2;ext2=$3; }

if [ ! -d "$dir" ] ; then
echo $usage
echo "$dir does not exist!"
exit
```

```

fi

if [ "$ext1" = "$ext2" ] ; then
    echo $usage
    echo "ext1 and ext2 must be different!"
    exit
fi

find $dir -name "*.$ext1" | while read file ; do
    base=`basename "$file" $ext1`
    mv -v "$file" "$base$ext2"
done

```

## 12.7 Sixth Exercise: Font Finder (\*)

**Commands used:** find, grep

**Synopsis:** fontfinder [directory] [ext]

**Description:** Write a script that finds font related informations in files with the extension 'ext'. Highlight the word 'fonts' and redirect error messages to /dev/null. The files should be searched in the given directory. If no directory is given, it should search in the current directory. If no extension is given, '\*.conf' should be used. Proper error handling should be done.

**Possible solution:** fontfinder.sh

```

#!/bin/sh

PARAMS=$#
ACTUALDIR='pwd'
EXT="conf"

if [ "$PARAMS" = "0" ]; then
true

elif [ "$PARAMS" = "1" ]; then

if [ -d "$1" ]; then
ACTUALDIR=$1
else
EXT=$1
fi

elif [ "$PARAMS" = "2" ]; then

if [ ! -d "$1" ]; then
echo "*** $1 *** is not a directory"
exit 1
fi

ACTUALDIR=$1

```

```
EXT=$2

fi

echo "> Info: Searching in $ACTUALDIR within files with extention $EXT"
find $ACTUALDIR -iname "*.${EXT}*" 2>/dev/null | grep -i --color fonts
```

## 12.8 Seventh Exercise: Size Sorter (\*\*)

**Commands used:** du, sort, tee, less

**Synopsis:** sizesorter [directory] [file]

**Description:** Write a script that creates a sorted list of the space used by the directories and files in the given directory. If no directory is given, the current directory should be searched. The result should be saved in the text file 'file'. After the file has been created, show it as a parallel and per page output on screen. Proper error handling should be done (e.g. the file already exists, etc.).

**Possible solution:** sizesorter.sh

```
#!/bin/sh

PARAMS=#
ACTUALDIR='pwd'
LOGFILE="${ACTUALDIR}/filesort.txt"

if [ "$PARAMS" = "0" ]; then

    echo "Usage: sizesort [directory] logfile"
exit 1

elif [ "$PARAMS" = "1" ]; then

if [ -d "$1" ]; then
    ACTUALDIR=$1
    elif [ ! -e "$1" ]; then
LOGFILE=$1
fi

elif [ "$PARAMS" = "2" ]; then

    if [ ! -d "$1" ]; then
        echo "*** $1 *** is not a directory"
        exit 1
    fi

LOGFILE=$2

fi
```

```
echo "> Info: Searching $ACTUALDIR and logging (APPENDING) to $LOGFILE"
du -sb ${ACTUALDIR}/* | sort -rn | tee $LOGFILE | less
```

## 12.9 Eighth Exercise: File Finder (\*\*)

**Commands used:** find, file

**Synopsis:** filefinder [script|image|program] [directory]

**Description:** Write a script that finds all files of a given type inside a given directory. If no directory is given, the current directory should be searched. The output should print the names of all found files. If the argument 'script' is given, only scripts should be found, if 'image' is given, only images should be found and if 'program' is given, only binary executables should be found. Proper error handling should be done.

**Possible solution:** filefinder.sh

```
#!/bin/bash

usage="usage: filefinder [script|image|program] [directory]"

cmd=$1
test -z "$2" && dir=. || dir=$2

case "$1" in
    script)
        echo "Finding scripts."
        pattern=$1
        ;;
    image)
        echo "Finding images."
        pattern=$1
        ;;
    program)
        echo "Finding programs."
        pattern="ELF"
        ;;
    "")
        dir=.
        echo "Finding all files."
        pattern=""
        ;;
    *)
        echo "Finding all files."
        pattern=""
        dir=$1
        ;;
esac
```

```

esac

test -d $dir || {
    echo $usage
    echo "Directory $dir does not exist!"
    exit 1
}

find $dir -type f | while read file ; do
    file "$file" | awk -F: '{print $2}' | grep -q "$pattern" && echo "$file"
done

```

## 12.10 Ninth Exercise: Find Web Server (\*\*\*)

**Commands used:** nmap

**Synopsis:** webfinder <list of IP addresses>|file

**Description:** Write a script that tests if there is a web server running on the given IP addresses. The script should test if the given IP addresses are reachable and if they respond to requests on port 80. It should output the result for each IP:

E.g.: 192.168.100.30 does not run a web sever  
 192.168.100.35 is not responding  
 192.168.100.254 runs a web sever

IP addresses can be either given as arguments on the command line or read from a file. Proper error handling should be done.

**Possible solution:** webfinder.sh

```

#!/bin/bash

usage="usage: webfinder <list of IP addresses>|file"

let "$# < 1" && { echo $usage; exit 1 ; }

( test -f "$1" && {
    cat "$1"
} || {
    echo $@ | tr ' ' '\n'
} ) | while read ip ; do

    ERRFILE=`mktemp`
    out=`nmap -p 80 $ip 2> $ERRFILE`
    grep -q 'Failed' $ERRFILE && echo "$1 is no valid IP address"
    echo "$out" | grep -q "tcp open" && echo "$ip runs a web server"
    echo "$out" | grep -q "tcp closed" && echo "$ip does not run a web server"
    echo "$out" | grep -q "down" && echo "$ip is not responding"

```

```
rm $ERRFILE 2>/dev/null
done
```

## 12.11 Tenth Exercise: Archiver (\*\*)

**Commands used:** tar, du

**Synopsis:** archiver [directory] size file

**Description:** Write a script that puts all files under a specified directory that are smaller than a defined size into an archive. If no directory is given, the current directory should be searched. Only files that are smaller than the 'size' argument should be included. The archive type should be 'tar.gz'. The name of the archive should be 'file'. Proper error handling should be done (e.g. no file specified).

**Possible solution:** archiver.sh

```
#!/bin/bash

usage="usage: archiver [directory] size file"

let "$# < 3" && { dir=.;size="$1";file="$2"; } ||
{ dir="$1";size="$2";file="$3"; }

if [ ! -d "$dir" -o -z "$size" -o -z "$file" ] ; then
  echo $usage
  exit 1
fi

[ -z "${size##[0-9]*}" ] || {
  echo "$size is not a number!"
  exit 1 ;
}

[ -e "$file" ] && file="$file`date +%s`"
echo "$file" | grep -q '.tar.gz' || file="$file.tar.gz"

files=
find $dir -type f | {
  while read a ; do
    fsize='du -b "$a" | cut -f1'
    [ "$fsize" -le "$size" ] && files="$files $a"
  done

  [ -n "$files" ] &&
  tar -z -v -c -f "$file" --exclude "$file" $files ||
  echo "No files found!"
}
```

## 12.12 Eleventh Exercise: Create svn home directory (\*)

**Commands used:** svn

**Synopsis:** svnhome [directory]

**Description:** Write a script that creates a directory named as the current user inside the svn directory given as an argument. If no directory is given, the current directory should be used. The created user directory should be committed with a proper commit message. Proper error handling should be done (e.g. directory already exists, given directory is not under svn control, etc.).

**Possible solution:** svnhome.sh

```
#!/bin/bash

usage="usage: svnhome [directory]"

let "$# < 1" && dir=. || dir="$1"

if [ ! -d "$dir" ] ; then
    echo $usage
    echo "$dir is not a directory!"
    exit 1
fi

if [ ! -d "$dir/.svn" ] ; then
    echo $usage
    echo "$dir is not a svn directory!"
    exit 1
fi

if [ -d "$dir/$USER" ] ; then
    echo $usage
    echo "An directory with your user name does already exist in $dir!"
    exit 1
fi

svn mkdir "$dir/$USER"
svn ci -m "Added user directory" "$dir/$USER"
```

## 12.13 Twelfth Exercise: Arping all 254 hosts in a given Class C network (\*)

**Commands used:** arping, arp

**Synopsis:** myping [network] [logfile]

**Description:** Write a script that does an arping on all 254 possible host in a class C network (e.g. 192.168.1) that should be passed as argument to the script. After that the script should

write the actual arp table into a log file. The log file should also be passed as argument.

**Possible solution:** myping.sh

```
#!/bin/sh

NETWORK=$1
INTERFACE=$2
LOGFILE=$3
INFO="No info yet"

for i in `seq 1 254`;
do
INFO='arping -c 1 -I $2 $1.$i 2>/dev/null';
echo "> INFO: $INFO"
done

echo "======" > $LOGFILE
arp >> $LOGFILE

exit 0
```

## **12.14 Thirteenth Exercise: Create a list of files that are opened by a process (\*\*\*)**

**Commands used:** lsof, grep, awk, du

**Synopsis:** plist [-p|-P] [pid|processname]

**Description:** Write a script that gets all files that are opened by a process and calculate the total size of these associated files. The output should be written to the stdout:

```
$ PROCESS: NAME
$ No. FILES: 40
$ TOTAL SIZE: 23MB
```

**Possible solution:** plist.sh

```
#!/bin/bash

usage="usage: plist [-p|-P] [pid|process_name]"

[ $# -gt 0 -a "$1" != "-p" -a "$1" != "-P" -o $# -eq 1 ] && {
echo $usage
exit 1
}
```

```
opt=$1
prog=$2
[ "$1" == "-P" ] && opt="-c"

lsof $opt $prog -F s | grep '^s' | sed 's/s//' | {
    sum=0
    while read num ; do
        let "sum+=$num"
        let "count++"
    done

    SUM=$sum; unit=
    let "$sum > 1024" && { let "SUM=$sum/1024"; unit="KB" ; }
    let "$sum > 1024*1024" && { let "SUM=$sum/(1024*1024)"; unit="MB" ; }

    [ -z "$prog" ] && prog=ALL
}

echo PROCESS:      $prog
echo No. FILES:   $count
echo TOTAL SIZE:  $SUM$unit
}
```

# C programming

---

## Contents

<b>13.1 Overview</b>	71
<b>13.2 Hello World</b>	71
<b>13.3 C preprocessor (cpp)</b>	72
<b>13.4 Keywords</b>	72
<b>13.5 man pages</b>	72

---

“C” is a general-purpose computer programming language. We will use C for system and network programming. That is the purpose for which C was developed in 1972 by Dennis Ritchie at the Bell Telephone Laboratories. Today C is in wide use for all kinds of purposes and on nearly every computing platform available. Many computing languages have been developed with “C-like” syntax.

## 13.1 Overview

“C” is an imperative programming language designed for system programming. Therefore it provides low-level access to memory and uses functions and data types that can run efficiently on computer hardware. When C-code is written standards-compliant, it can be compiled and run on many different hardware platforms with very little porting effort.

## 13.2 Hello World

Let’s start with a simple example program: the classical “hello world” program. (By the way: the “hello world” example was first published by Kernighan and Ritchie, the authors of “The C Programming Language” – long time the informal specification of the C language.)

```
#include <stdio.h>

int main(int argc, char **argv) {
    printf("Hello World\n");

    return 0;
}
```

The first line of the program contains a preprocessing directive (`#include <stdio.h>`). This causes the preprocessor – the first tool to examine source code as it is compiled – to substitute the line with the entire text of the `stdio.h` standard header, which contains declarations for standard input and output functions such as `printf`. In Linux the standard headers are located in `/usr/include`.

The next line (`int main(int argc, char **argv) {}`) starts a function. The first word is the type of the return value (`integer`) of the function, followed by the functions name (`main`) and the arguments to it in parenthesis (`(int argc, char **argv)`). A function receives values through arguments. In this case it receives two values: an integer and a “list of strings” (an array of an array of characters). The functions code begins with the opening braces, and ends with the closing braces four lines below.

The “`main`” function is special, in that it is the starting point for the execution of every C-program.

The following line (`printf("Hello World\n");`) uses a function defined in `stdio.h` to format and print its argument(s) to standard out – usually a console / terminal – using a system library.

The line “`return 0;`” terminates the execution of the `main` function and returns 0 as its “exit code” – which is interpreted by the run-time system as “success”.

### 13.3 C preprocessor (cpp)

The C preprocessor (cpp) is the preprocessor for the C programming language. In many C implementations, it is a separate program invoked by the compiler as the first part of translation. The preprocessor handles directives for source file inclusion (`#include`), macro definitions (`#define`), and conditional inclusion (`#if`).

### 13.4 Keywords

The following words are keywords in the C-language<sup>1</sup>. Those words are reserved and must not be used as variable or function names: `auto`, `break`, `case`, `char`, `const`, `continue`, `default`, `do`, `double`, `else`, `enum`, `extern`, `float`, `for`, `goto`, `if`, `int`, `long`, `register`, `return`, `short`, `signed`, `sizeof`, `static`, `struct`, `switch`, `typedef`, `union`, `unsigned`, `void`, `volatile`, `while`.

### 13.5 man pages

With C being the main programming language for UNIX system programming, most UNIX systems package a C reference manual with the operating system. For Linux the manual can be read by using the `man` pages:

---

<sup>1</sup>This list is for ANSI-C, other C-standards may have additional keywords.

```
$ man -S 3 printf
```

You can also use `man 3 printf`.

All man(ual) pages follow a common layout that is optimized for presentation on a simple ASCII text display, possibly without any form of highlighting or font control. Sections present may include:

NAME	The name of the command or function, followed by a one-line description of what it does.
SYNOPSIS	In the case of a command, you get a formal description of how to run it and what command line options it takes. For program functions, a list of the parameters the function takes and which header file contains its definition. For experienced users, this may be all the documentation they need.
DESCRIPTION	A textual description of the functioning of the command or function.
EXAMPLES	Some examples of common usage.
SEE ALSO	A list of related commands or functions.

Other sections may be present, but these are not well standardized across man pages. Common examples include: OPTIONS, EXIT STATUS, ENVIRONMENT, KNOWN BUGS, FILES, AUTHOR, REPORTING BUGS, HISTORY and COPYRIGHT.

When you look at the `printf` man page you will notice, that it also contains the description for other related functions and it tells you which headers must be included to use the function(s).



# Basic Networking

---

## Contents

<b>14.1 What is Networking? . . . . .</b>	<b>75</b>
<b>14.2 The four layers of TCP/IP . . . . .</b>	<b>76</b>
14.2.1 The Link Layer (lowest) . . . . .	76
14.2.2 The IP Layer . . . . .	77
14.2.3 The Transport layer . . . . .	77
14.2.4 The Application layer (highest) . . . . .	77
<b>14.3 The MAC and the IP(v4) Address . . . . .</b>	<b>77</b>
14.3.1 IANA-reserved private IPv4 network ranges . . . . .	78
14.3.2 There is no place like 127.0.0.1 . . . . .	78
<b>14.4 What Configuration has to be done to get access to the Internet? . . . . .</b>	<b>79</b>
14.4.1 MAC Address . . . . .	79
14.4.2 IP Address/Subnetmask . . . . .	79
14.4.3 (Default) Gateway . . . . .	79
14.4.4 DNS Server . . . . .	79
14.4.5 Manual configuration on the fly . . . . .	79
14.4.6 Manual configuration . . . . .	80
14.4.7 Configuration with DHCP per default . . . . .	80
14.4.8 Illustrated Connection with DHCP . . . . .	82
<b>14.5 Network connection: trouble shooting . . . . .</b>	<b>83</b>

Almost every Computer nowadays is connected to other computers. Such networks allow computers to communicate with each other and share resources and information.

This chapter gives a brief overview how computers communicate using the IP layer of the TCP/IP-stack. It provides basic knowledge how to configure hosts for communication in simple IPv4 networks.

It does not explain the details of the whole TCP/IP stack, the upcoming IPv6 or concepts of efficient routing.

## 14.1 What is Networking?

A computer network is a collection of computers and devices connected to each other. The network allows computers to communicate with each other and share resources and information.

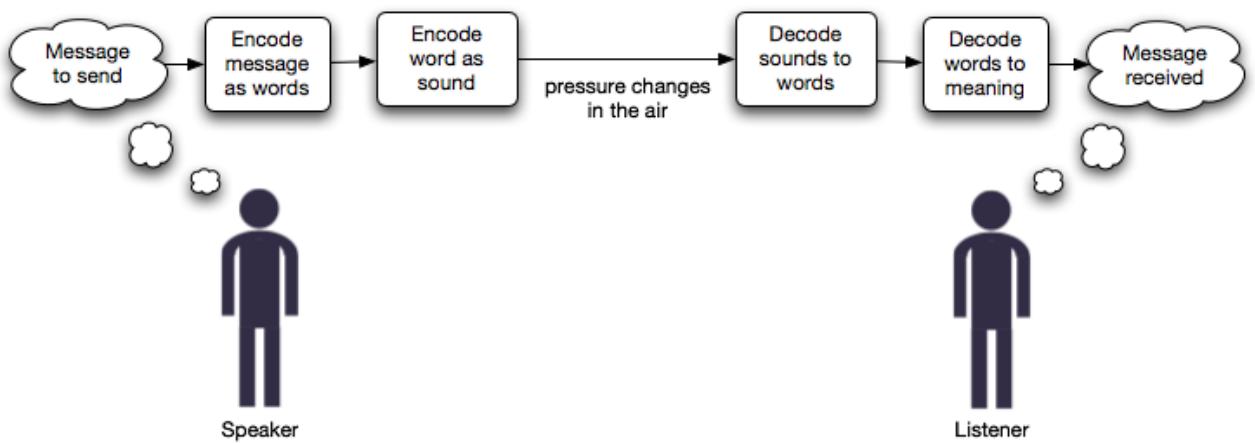


Figure 14.1: (one way) communication

We need a code that both sender and receiver are able to understand, and a medium to transmit. In this case (spoken word) this is a common language as code, and mouth and ears (and air between) as media.

Another example would be to send someone a letter. In this case, the code is written language, and the media are paper, letterbox, postman, post office, car, plane, car, post office, postman, letter box. It is not enough to know how to write, to send a letter, but it is also necessary to know the protocol to address, stamp and send a letter. You do not have to know how to fly a plane however, as everything behind the letterbox where you put your letter in is hidden from you.

If it comes to computers, there is hardware (medium) and protocols (codes, "languages"). There are plenty of protocols for networking (and Linux), most important for us now is the TCP/IP suite of protocols, famous for being the protocols that "the Internet" speaks.

The TCP/IP Suite uses encapsulation, which means that it has four layers for different tasks, with each layer just transporting every (protocol) data of the higher layers with just caring for its own protocol.

Just like you sending a letter to Iraq without knowing how to fly a plane, and without the pilot knowing the street in Baghdad where the addressed house has its mailbox. In this case you would be seen as the Application (layer), the envelope as the Transportation layer, the (knowledge of the) chain of post offices as the Internet layer, and post bikes, cars, planes then as the Link layer.

## 14.2 The four layers of TCP/IP

### 14.2.1 The Link Layer (lowest)

The link layer is responsible for "one hop" from one machine to the next gateway or to the target machine if the target machine is in the same subnet. Protocols: Ethernet, PPP, ARP et al. Important Elements for building up a basic connection: **MAC address**

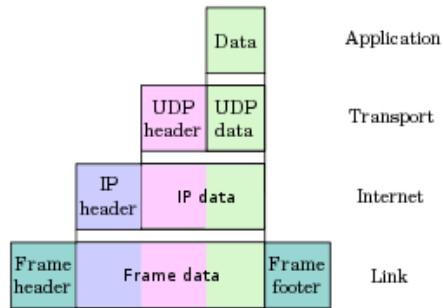


Figure 14.2: encapsulation

#### 14.2.2 The IP Layer

The IP layer is responsible for the Routing

Important Elements for building up a basic connection: **IP address**, **subnetmask**, (default) **gateway**

#### 14.2.3 The Transport layer

The transport layer is responsible for controlling the (re)transmission.

Important Elements for building up a basic connection: **port** (socket)

#### 14.2.4 The Application layer (highest)

The application layer consists of the protocol ("language") actually "spoken" (HTTP, SMTP etc.) by client and server applications.

Important for building up a basic connection: **DNS**, **DHCP**

### 14.3 The MAC and the IP(v4) Address

The MAC address is a unique identifier in the form of a 12-digit hexadecimal number (for example 00:DE:AD:BE:EF:00) hard-coded to any (Ethernet) networking device by the manufacturer. The only way to send information to a computer is to this address. However, as network equipment is shipped all over the world it is impossible to know where a certain MAC address is located. Therefore it is only possible to send information to MAC addresses that are in the same subnet, as you are only able to reach those machines via broadcast.

A subnet is a group of computers that are on the same link, so that every computer is reachable from another in one "hop", with no other machine (router) in between (switches don't count). A broadcast (message) is a message sent to all computers on the link, i.e. all computers in the subnet.

For reaching computers that are not directly connected to us (i.e. are not in the same subnet),

we need another way of addressing those, the IP address. The process of getting information delivered into another subnet is called "routing" (see chapter 15 on page 87). However, the IP protocol is also used for communication with machines in the same subnet and the loopback device.

While the MAC address comes with the hardware, the IP address has to be assigned. An IP address is technically a 32-bit (= 4 bytes) binary number. However, the four bytes are normally displayed in decimal numbers, separated by a dot.

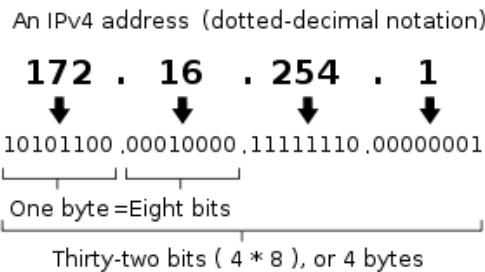


Figure 14.3: IP Address

#### 14.3.1 IANA-reserved private IPv4 network ranges

	Start	End	No. of addresses
24-bit Block (8 prefix, 1 x A)	10.0.0.0	10.255.255.255	16,777,216
20-bit Block (12 prefix, 16 x B)	172.16.0.0	172.31.255.255	1,048,576
16-bit Block (16 prefix, 256 x C)	192.168.0.0	192.168.255.255	65,536

The IP addresses of these *private* IP ranges are special in the way that they cannot be reached from the internet (they are called "non-routable addresses"). Therefore you are able to assign them in your own network without any conflicts to other machines on the internet without any coordination. To connect computers with private IP addresses to the Internet, you will most likely have a router with a public IP address doing NAT. A direct connection is not possible.

#### 14.3.2 There is no place like 127.0.0.1

127.x.x.x is reserved for your loopback device, which is used by the computer to do TCP/IP communication with itself. Of course this is not attached to a network card, it is a virtual device and every computer "speaking" TCP/IP will have one. The name "localhost" usually resolves to 127.0.0.1. Always remember: This is **you!**

## 14.4 What Configuration has to be done to get access to the Internet?

### 14.4.1 MAC Address

The MAC address is preconfigured by the manufacturer of your NetworkInterfaceCard. If you have a NIC installed

```
$ ifconfig -a  
or  
$ ip link
```

will show it. If nothing is shown, your NIC is not ready to use (hardware broken or correct drivers not loaded).

### 14.4.2 IP Address/Subnetmask

There are basically two ways of getting an IP address / netmask. You can configure one manually, or you use a special service called DHCP that will do this (dynamic) configuration automatically.

### 14.4.3 (Default) Gateway

A gateway is a device (router) connecting (at least) two networks, ultimately connecting you to the internet. You can configure your default gateway manually, or you use a special service called DHCP that will do this (dynamic) configuration automatically<sup>1</sup>.

### 14.4.4 DNS Server

Decimal numbers are more difficult to remember than common names. The DomainNameService (DNS) provides a mapping from names to IP addresses. If you want to use [www.yahoo.de](http://www.yahoo.de) instead of 87.248.120.129, you will need a DNS Server who resolves this name to an IP address. You can configure one manually on the computer, or you use a special service called DHCP that will do this (dynamic) configuration automatically.

### 14.4.5 Manual configuration on the fly

Setting the IP address for network interface card eth0:

---

<sup>1</sup>See chapter 15 on page 87

```
$ ifconfig eth0 192.168.0.66/24
or
$ ip addr add 192.168.0.66/24 dev eth0
```

Note that those changes will be gone after a restart.

Setting the default gateway (if the router has 192.168.0.1 see also chapter 15 on page 87):

```
$ sudo route add default gw 192.168.0.1 dev eth0
or
$ sudo ip route add default via 192.168.0.1 dev eth0
```

Note that those changes will be gone after a restart.

To setup your DNS configuration, edit `/etc/resolv.conf` like this (if the IP address of your nameserver is 130.149.4.20):

```
nameserver 130.149.4.20
```

#### 14.4.6 Manual configuration

If the configuration should be persistent after reboot, edit the networking configuration files of your Linux distribution. For Debian/Ubuntu, `/etc/network/interfaces` has to be edited like that:

```
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.66
    gateway 192.168.0.1
    netmask 255.255.255.0
    network 192.168.0.0
    broadcast 192.168.0.255
```

#### 14.4.7 Configuration with DHCP per default

In case of using DHCP service, edit `/etc/network/interfaces` like this:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface - use DHCP to find our address
auto eth0
iface eth0 inet dhcp
```

#### 14.4.8 Illustrated Connection with DHCP

First we plug the computer into the network jack, then the DHCP client broadcasts for a DHCP service:

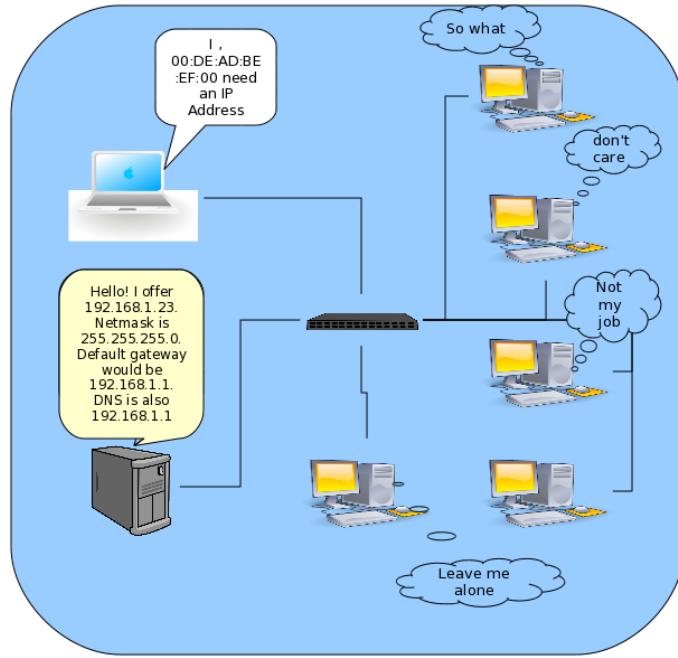


Figure 14.4: connected and crying (broadcasting) for help (dhcp)

Now we connect to a computer in our own subnet:

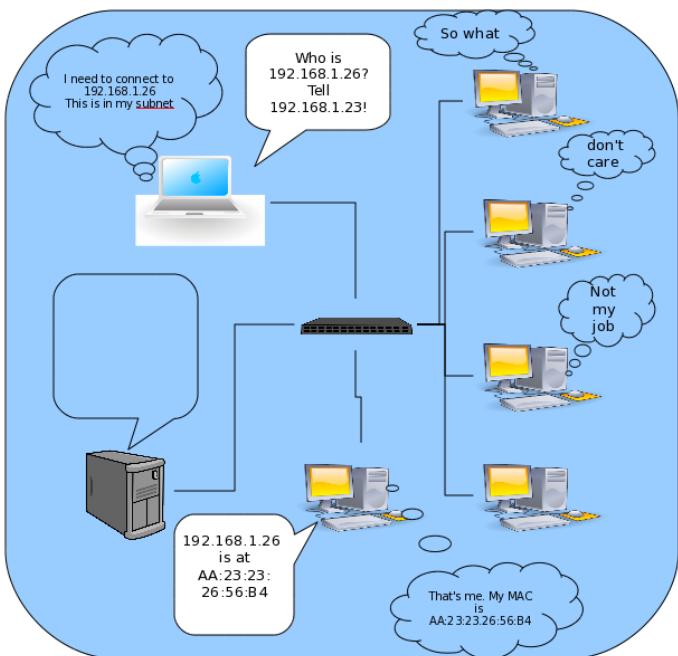


Figure 14.5: configured and shouting (broadcasting) for MAC Address

Now we connect to a computer not located in our own subnet:

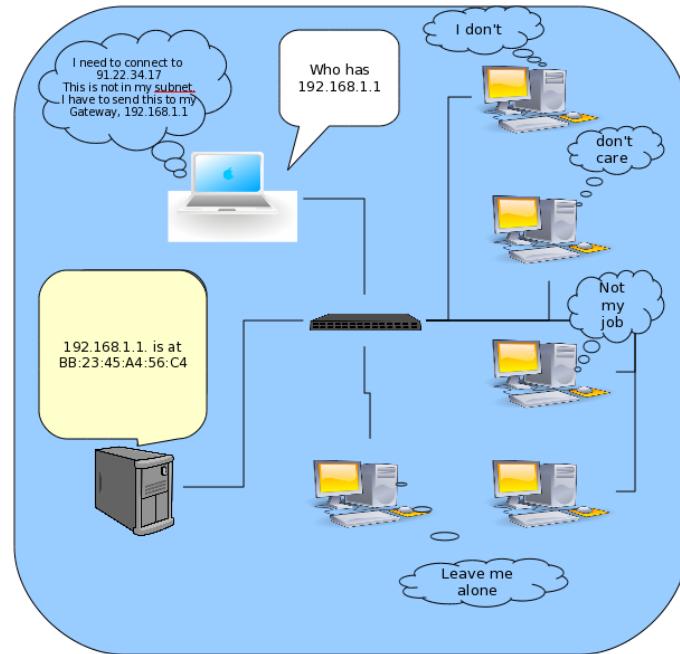


Figure 14.6: everything going to outside the subnet is send to the (default) gateway

## 14.5 Network connection: trouble shooting

If there is something wrong with the internet perform the following steps (ether from top to bottom or from bottom to top)

- cable:

```
$ ethtool eth0
$ mii-tool
```

wireless:

```
$ iwconfig
```

check plugs

- ip:

check ip

```
$ ifconfig  
or  
$ ip addr
```

ping ip local router

```
$ ping 192.168.1.1
```

- route:

check route

```
$ route -n  
or  
# ip route show
```

ping an ip in the internet (for instance of a dns server)

```
$ ping 130.149.4.20
```

- dns:

ping url

```
$ ping google.com
```

check resolv.conf

```
$ sudo nano /etc/resolv.conf
```

- application (ex. firefox):

check whether it is up and running, logfiles? (ex. firefox: check File->Work Offline)

- target computer (server)

Is the target up?:

```
$ ping targethostname
```

Is the target's service port listening? Is the service listening on that port the service that is expected?

```
$ telnet target port  
$ nmap -A -p port targethostname
```



# Routing

---

## Contents

<b>15.1 Overview</b>	87
<b>15.2 Configuring the kernel routing table</b>	87
15.2.1 Showing the kernel routing table	88
15.2.2 Adding a route	89
15.2.3 Removing a route	89
<b>15.3 The default route</b>	89
<b>15.4 Behind the scenes</b>	90

---

Routing is necessary, whenever two (or more) networks should be connected. Networks are connected through “routers” by attaching each network to a different network interfaces of the router. Routers decide where to send each packet to, depending on its destination. The decision is based on a rule set - the “routing table”.

## 15.1 Overview

Small networks may involve manually configured routing tables: *static routing*, while larger networks involve complex topologies and may change rapidly, making the manual construction of routing tables unfeasible. *Adaptive routing* attempts to solve this problem by constructing routing tables automatically, based on information carried by routing protocols, and allowing the network to act nearly autonomously in avoiding network failures and blockages.

*Dynamic routing* dominates the Internet. However, the configuration of the routing protocols often requires a skilled touch; one should not suppose that networking technology has developed to the point of the complete automation of routing.

We are going to take a look only at static routing.

## 15.2 Configuring the kernel routing table

For each packet the kernel wants to send, it goes through all the rules in the routing table *from top to bottom*, until the packets destination IP matches a rule. Thus, the order of the rules is important!

If no rule matched the packets destination, then the error “Destination Host Unreachable” is returned.

To configure the kernel (operating system) routing table you can use the classical **route** command or the newer **ip** command. Both are fully supported on all Linux platforms, but most people still use the “route” command.

### 15.2.1 Showing the kernel routing table

To show the kernel routing table in Linux, run **route -n** or **ip route show**:

```
$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
192.168.20.0   0.0.0.0        255.255.255.0 U      1      0    0    eth0
127.0.0.0      127.0.0.1      255.0.0.0     UG     0      0    0    lo
0.0.0.0         192.168.20.1   0.0.0.0       UG     0      0    0    eth0

$ /sbin/ip route show
192.168.20.0/24          dev  eth0  proto kernel scope link \
                           src 192.168.20.227 metric 1
127.0.0.0/8              via 127.0.0.1  dev  lo
default                   via 192.168.20.1  dev  eth0  proto static
```

To show the kernel routing table in Windows XP, run **route print**:

```
C:\> route print
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x2 ...08 00 27 da d6 2c .... AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
=====
=====
Active Routes:
Network Destination Netmask Gateway Interface Metric
0.0.0.0 0.0.0.0 192.168.20.1 192.168.20.117 20
127.0.0.0 255.0.0.0 127.0.0.1 127.0.0.1 1
192.168.20.0 255.255.252.0 192.168.20.117 192.168.20.117 20
192.168.20.117 255.255.255.255 127.0.0.1 127.0.0.1 20
192.168.20.255 255.255.255.255 192.168.20.117 192.168.20.117 20
224.0.0.0 240.0.0.0 192.168.20.117 192.168.20.117 20
255.255.255.255 255.255.255.255 192.168.20.117 192.168.20.117 1
Default Gateway: 192.168.20.1
=====
Persistent Routes:
None
```

### 15.2.2 Adding a route

If we want to connect to the 192.168.30.0/24 network through a router at the IP 192.168.20.12, we have to execute:

```
$ sudo route add -net 192.168.30.0/24 gw 192.168.20.12 dev eth0
or
$ sudo ip route add 192.168.30.0/24 via 192.168.20.12 dev eth0
```

No output will be shown if the commands execute successfully.

```
$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags   Metric  Ref  Use   Iface
 192.168.30.0   192.168.20.12  255.255.255.0  UG      0       0    0    eth0
 192.168.20.0   0.0.0.0        255.255.255.0  U        1       0    0    eth0
 127.0.0.0      127.0.0.1      255.0.0.0      UG      0       0    0    lo
 0.0.0.0        192.168.20.1   0.0.0.0        UG      0       0    0    eth0

$ /sbin/ip route show
192.168.30.0/24  via 192.168.20.12  dev  eth0
192.168.20.0/24            dev  eth0  proto kernel scope link \
                           src 192.168.20.227 metric 1
127.0.0.0/8        via 127.0.0.1      dev  lo
default          via 192.168.20.1    dev  eth0  proto static
```

For example, if the kernel wants to send a packet to the host 192.168.20.4, the first rule of the routing table above would not match. It does only match packets for the IP addresses 192.168.30.1 to 192.168.30.254. The second rule does match, because IP addresses 192.168.20.1 to 192.168.20.254 are matched (and that includes 192.168.20.4).

### 15.2.3 Removing a route

To remove this rule from the routing table run the same command, but with “del” instead of “add”:

```
$ sudo route del -net 192.168.30.0/24 gw 192.168.20.12 dev eth0
or
$ sudo ip route del 192.168.30.0/24 via 192.168.20.12 dev eth0
```

## 15.3 The default route

For all networks that can be connected to directly through a router in your own LAN, you add a route. But what about other networks? What about the internet?

For all these other networks, for which you don't know how to connect to, you add the "default route". That is a rule, that is always put at the bottom of the routing table, and that matches **every** destination. This way, all packets that did not match a specific rule<sup>1</sup>, will be send to the "default gateway".

The "default gateway" typically is a normal router, that knows other networks and has routes to them. It also has a default route (through a default gateway) for all networks it does not connect to directly.

To add a "default route" you run the same command as for adding normal routes, just adding the keyword "default" in the right place.

But before adding a default route, you'll have to remove the old default route first<sup>2</sup>:

```
$ sudo route del default gw 192.168.20.1
or
$ sudo ip route del default
```

To add a default route run:

```
$ sudo route add default gw 192.168.20.50 dev eth0
or
$ sudo ip route add default via 192.168.20.50 dev eth0
```

Your new routing table looks like this:

```
$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.30.0   192.168.20.12  255.255.255.0  UG      0      0      0      eth0
192.168.20.0   0.0.0.0        255.255.255.0  U        1      0      0      eth0
127.0.0.0      127.0.0.1      255.0.0.0      UG      0      0      0      lo
0.0.0.0         192.168.20.50  0.0.0.0       UG      0      0      0      eth0
```

## 15.4 Behind the scenes

Routers work on layer 3 of the ISO/OSI model<sup>3</sup> ("Network Layer"). The only information they base their routing decisions on are the destination IP of each packet. They never change this information (and neither do they change the senders IP).

---

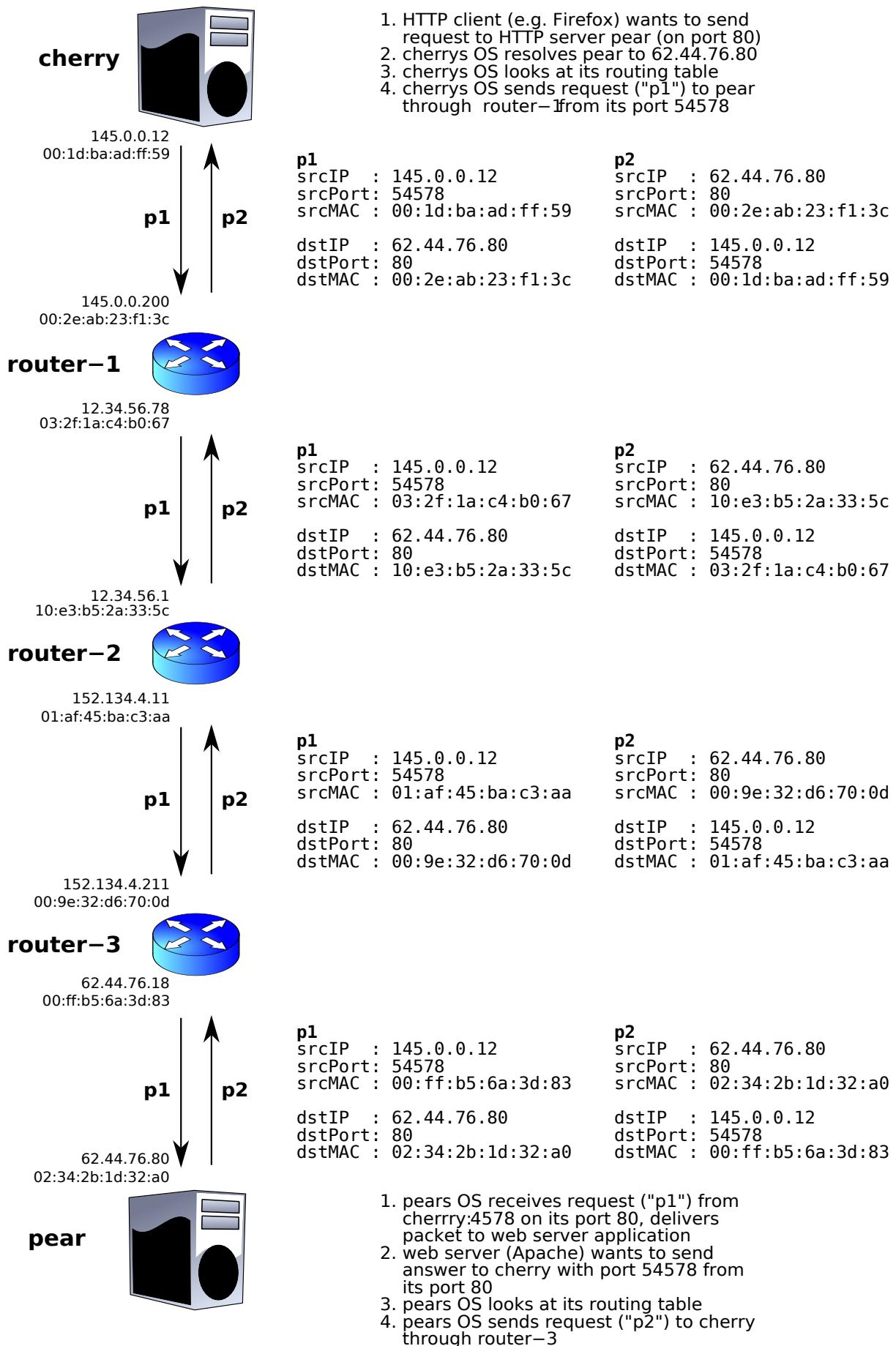
<sup>1</sup>Remember that rules are read from top to bottom.

<sup>2</sup>I am assuming 192.168.20.1 was the previous default gateway. Check your own environment by running "route -n".

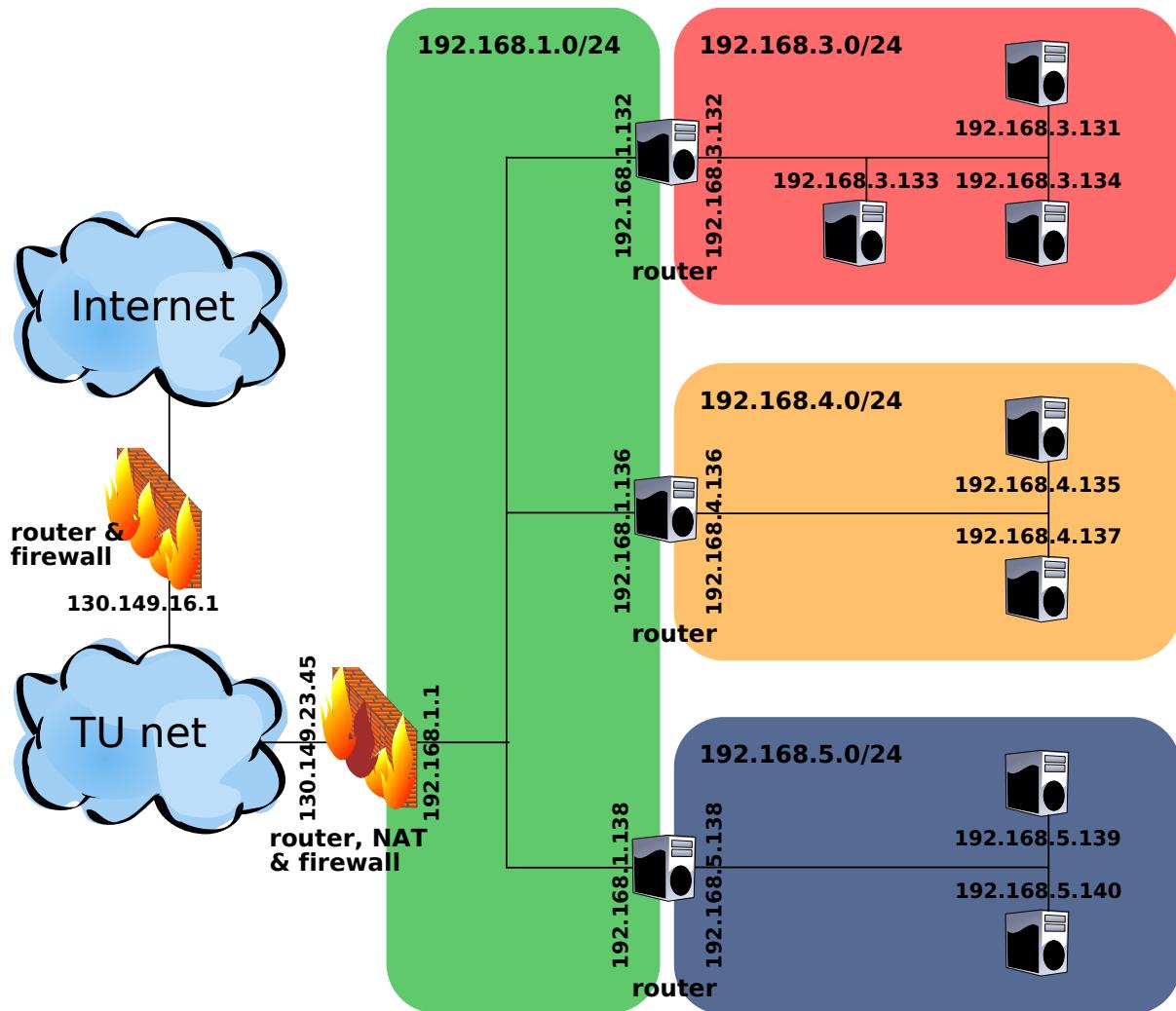
<sup>3</sup>see [http://www.thecertificationhub.com/networkplus/the\\_osi\\_ref\\_model.htm](http://www.thecertificationhub.com/networkplus/the_osi_ref_model.htm)

To send a packet to the final host or another intermediate router they must set the sending physical address (MAC) to their own MAC, and the destination MAC to the recipients MAC. These are the only changes routers make to packets.

The following diagram shows a client (“cherry”) requesting a web page from a server (“pear”). The request and the response packets pass several routers (“router-1”, “router-2” and “router-3”) on their way. These routers do only change the MAC addresses of the packets they forward.



The following diagram shows an example network setup. Each of the four routers (incl. the firewall) in the 192.168.1.0/24 network must have routes to each of the other networks. The firewall needs routes to all three networks too, so it can return answers to requests sent out to the internet.



All PCs in the 192.168.3.0/24, 192.168.4.0/24 and 192.168.5.0/24 networks could have routes to all other networks. But it is easier (and better maintainable) to just configure all routes on the routers, and to configure the PCs to send all traffic to them (see “default gateway”). The routers will then forward all packets to the correct networks through the corresponding routers.

The router for the 192.168.3.0/24 network would create its routes like this:

```
$ route add -net 192.168.4.0/24 gw 192.168.1.136 dev eth0
$ route add -net 192.168.5.0/24 gw 192.168.1.138 dev eth0
$ route add default gw 192.168.1.1 dev eth0
```



# Routing Exercises

---

## Contents

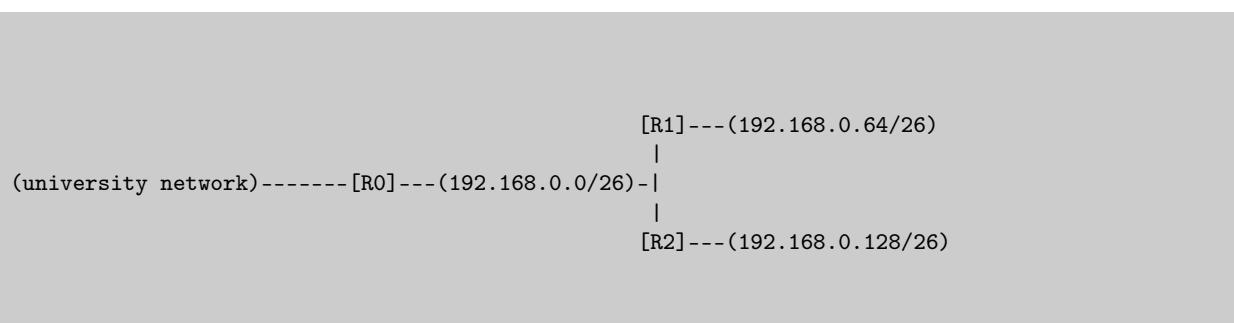
<b>16.1 Scenario</b>	95
16.1.1 Preparing the routing machines	96
16.1.2 Configure Network Interfaces: Router 1, Router 2	96
16.1.3 Enabling IP Forwarding for IPv4: Router 1, Router 2	96
16.1.4 Configure Network Interfaces: Router 0	98
16.1.5 Outcome: The configured interfaces	98
16.1.6 Outcome: The established routes	98
16.1.7 Outcome: Setting up the clients	98
<b>16.2 Exercise: Network Redesign (Optional)</b>	99
<b>16.3 Exercise: Implementing firewall rules</b>	100
<b>16.4 Refine Network Topology</b>	101
<b>16.5 Exercise: FW persistence</b>	102
<b>16.6 Transfer</b>	103

---

In the following we want to apply the theories of routing in several different scenarios.

## 16.1 Scenario

Network topology with one class C network Setting up three (three of four possible) sub networks defined by subnetmask 26 192.168.1.0/26 (net0, outer network), 192.168.2.0/26 and 196.168.3.0/26 (net1 & net2, inner networks)



### 16.1.1 Preparing the routing machines

Installing two network interface adapters

Installing debian minimal system (from netinstaller cd e.g.)

Installing package resolveconf for being able to configure DNS from interfaces file

```
/etc/network/interfaces  
[...]  
  
#dns information  
dns-nameservers 192.168.0.1  
[...]
```

### 16.1.2 Configure Network Interfaces: Router 1, Router 2

Setting up eth0 as external (to net0) interface

Setting up eth1 as internal (to net1 respectively net1) interface

Router 1: eth1 192.168.0.65/26, Router 2: eth1 192.168.0.129/26

Adding the particular route, each to the other network

Router 1: route add -net 192.168.0.128/26 netmask 255.255.255.192 gw 192.168.0.61

Router 2: route add -net 192.168.0.64/26 netmask 255.255.255.192 gw 192.168.0.60

Adding name server information to /etc/network/interfaces (requires resolvconf package)

( compare figure 16.1 )

### 16.1.3 Enabling IP Forwarding for IPv4: Router 1, Router 2

IP forwarding has to be enabled explicitly. To make the configuration persistent we add that information to sysctl.conf file. ( compare figure 16.2 )

```
/etc/sysctl.conf  
[...]  
  
net.ipv4.ip_forward = 1  
[...]
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.61
    netmask 255.255.255.192
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1
    post-up route add -net 192.168.0.128 netmask 255.255.255.192 gw 192.168.0.61
    pre-down route del -net 192.168.0.128 netmask 255.255.255.192 gw 192.168.0.61

auto eth1
iface eth1 inet static
    address 192.168.0.65
    netmask 255.255.255.192
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1
    post-up route add -net 192.168.0.64 netmask 255.255.255.192 gw 192.168.0.64
    pre-down route del -net 192.168.0.64 netmask 255.255.255.192 gw 192.168.0.64

# /etc/network/interfaces" 21L, 597C
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.61
    netmask 255.255.255.192
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1
    post-up route add -net 192.168.0.128 netmask 255.255.255.192 gw 192.168.0.61
    pre-down route del -net 192.168.0.128 netmask 255.255.255.192 gw 192.168.0.61

auto eth1
iface eth1 inet static
    address 192.168.0.129
    netmask 255.255.255.192
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1
    post-up route add -net 192.168.0.64 netmask 255.255.255.192 gw 192.168.0.64
    pre-down route del -net 192.168.0.64 netmask 255.255.255.192 gw 192.168.0.64

# /etc/network/interfaces" 22L, 597C
```

Figure 16.1: Network configuration of both routers

```
#net.ipv4.tcp_syncookies=1
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
# Uncomment the next line to enable packet forwarding for IPv6
net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Ignore ICMP broadcasts
net.ipv4.icmp_ignore_broadcasts = 1
#
# Ignore bogus ICMP errors
net.ipv4.icmp_ignore_bogus_error_responses = 1
#
# Do not accept ICMP redirects (prevent MITM attacks)
net.ipv4.conf.all.accept_redirects = 0
```

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.0.1
```

Figure 16.2: IP forwarding, domain name resolution

```

debian-router:~# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:1c:42:cf:84:1c
          inet addr:192.168.0.60  Bcast:192.168.0.63  Mask:255.255.255.192
          inet6 addr: fe80::21c:42ff:fe84:1c%eth0  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2333 (2.2 Kib)  TX bytes:2088 (2.0 Kib)
          Interrupt:10 Base address:0x8200

debian-router:~# ifconfig eth1
eth1      Link encap:Ethernet HWaddr 00:1c:42:ed:8e:c3
          inet addr:192.168.0.65  Bcast:192.168.0.127  Mask:255.255.255.192
          inet6 addr: fe80::21c:42ff:feed:8e3%eth1  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:552 (552.0 B)  TX bytes:468 (468.0 B)
          Interrupt:10 Base address:0x8400

debian-router:~# _
```

```

debian-router-2:~# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:1c:42:22:7e:ea
          inet addr:192.168.0.61  Bcast:192.168.0.63  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:26 errors:8 dropped:8 overruns:8 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1853 (1.8 Kib)  TX bytes:1440 (1.4 Kib)
          Interrupt:10 Base address:0x8200

debian-router-2:~# ifconfig eth1
eth1      Link encap:Ethernet HWaddr 00:1c:42:ff:18:00
          inet addr:192.168.0.129  Bcast:192.168.0.191  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:552 (552.0 B)  TX bytes:0 (0.0 B)
          Interrupt:10 Base address:0x8400

debian-router-2:~# _
```

Figure 16.3: interfaces of both routers

#### 16.1.4 Configure Network Interfaces: Router 0

Setting up eth0 as external (to university gateway) interface

Setting up eth1 as internal (to net1) interface

Adding the particular routes, to each of the internal networks:

```
route add -net 192.168.0.128/26 netmask 255.255.255.192 gw 192.168.0.61
route add -net 192.168.0.64/26 netmask 255.255.255.192 gw 192.168.0.60
```

#### 16.1.5 Outcome: The configured interfaces

The interfaces eth0 and eth1 of Router 1 respectively Router 2 have to be configured like shown in figure 16.3. eth0 targets the external network where eth1 is related to the internal one (net1).

#### 16.1.6 Outcome: The established routes

The following routing table is established on Router 1 respectively Router 2 ( compare figure 16.4 ). Both routers have information of how to route into own networks, into each other private networks and - per default - into external networks.

#### 16.1.7 Outcome: Setting up the clients

If client are located each in one of these (sub)network the interfaces have to be configured like illustrated in figure 16.5 to be able to reach each other and to establish connections to external networks.

The screenshot shows two terminal windows side-by-side. The left window is titled "Debian Router 1 – Parallels Desktop" and the right window is titled "Debian Router 2 – Parallels Desktop". Both windows display the output of the command "route -n".

```
debain-router-1:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
192.168.0.0     0.0.0.0       255.255.255.192 U     0      0 eth0
192.168.0.64    0.0.0.0       255.255.255.192 U     0      0 eth1
192.168.0.128   192.168.0.61  255.255.255.192 UG    0      0 eth0
0.0.0.0         192.168.0.1   0.0.0.0       UG    0      0 eth0
debain-router-1:~# _
```

```
debain-router-2:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
192.168.0.0     0.0.0.0       255.255.255.192 U     0      0 eth0
192.168.0.64    192.168.0.60  255.255.255.192 UG    0      0 eth0
192.168.0.128   0.0.0.0       255.255.255.192 U     0      0 eth1
0.0.0.0         192.168.0.1   0.0.0.0       UG    0      0 eth0
debain-router-2:~# _
```

Figure 16.4: routing information of both routers

The screenshot shows two terminal windows side-by-side. The left window is titled "Debian Client 1 – Parallels Desktop" and the right window is titled "Debian Client 2 – Parallels Desktop". Both windows display the output of the command "ifconfig eth0".

```
debian-client-1:~# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:1c:42:e4:b8:ec
          inet addr:192.168.0.78  Bcast:192.168.0.127  Mask:255.255.255.128
          inet6 addr: fe00::21c:42ff:fe4:b8ec/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1064 (1.0 Kib)  TX bytes:1248 (1.2 Kib)
          Interrupt:10 Base address:0xb200
debian-client-1:~# _
```

```
debian-client-2:~# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:1c:42:ef:93:33
          inet addr:192.168.0.149  Bcast:192.168.0.191  Mask:255.255.255.192
          inet6 addr: fe00::21c:42ff:feef:9333/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1004 (1.0 Kib)  TX bytes:1248 (1.2 Kib)
          Interrupt:10 Base address:0xb200
debian-client-2:~# _
```

Figure 16.5: interfaces of clients located in both subnetworks

Figure 16.6 finally describes the routing tables of clients that are located in net1 respectively net2.

## 16.2 Exercise: Network Redesign (Optional)

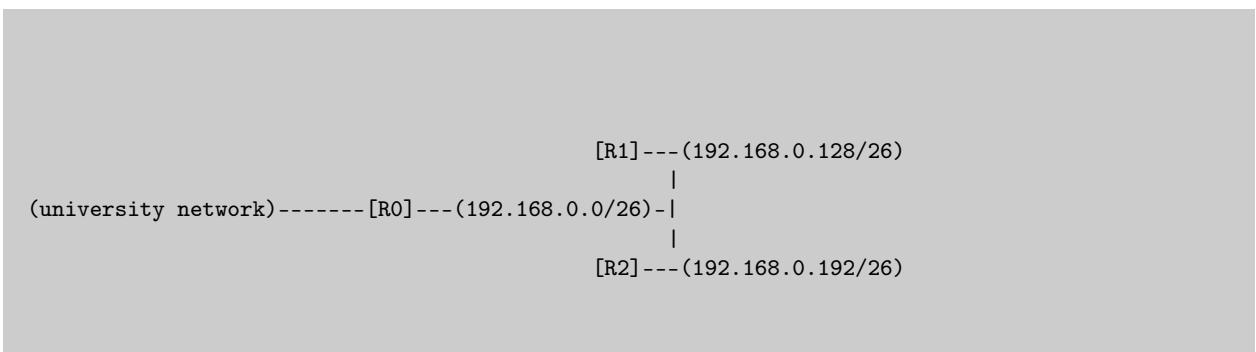
Now, the network setting change according to the given structure. The IP addresses of both routers respectively of both (sub)networks have changed. Adjust the settings according to the previous procedure for involved routers and for clients that are located in the (sub)networks. (interface settings of router 1, router 2; routing information of router 1, router 2; configuration of clients located in (sub)networks)

The screenshot shows two terminal windows side-by-side. The left window is titled "Debian Client 1 - Parallels Desktop" and the right window is titled "Debian Client 2 - Parallels Desktop". Both windows display the output of the command "route -n".

```
debain-client-1:# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
192.168.0.64   0.0.0.0       255.255.255.192 U     0      0      0 eth0
0.0.0.0         192.168.0.65  0.0.0.0       UG    0      0      0 eth0
debain-client-1:# -
```

```
debain-client-2:# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
192.168.0.128  0.0.0.0       255.255.255.192 U     0      0      0 eth0
0.0.0.0         192.168.0.129 0.0.0.0       UG    0      0      0 eth0
debain-client-2:# -
```

Figure 16.6: routing information of clients located in both subnetworks



Setup ALL clients (50:50) into the subnetworks

### 16.3 Exercise: Implementing firewall rules

Setting up iptables rules to implement the following constraints. Where you have to put each of these rules?

Policy: ACCEPT, so we have to define what to REJECT and where we have to define exceptions (see last example, from very special to more general)

```
net2 -|<- net1 ( Router 1, FORWARD )

iptables -A FORWARD -p tcp -s 192.168.0.128/26 -d 192.168.0.64/26 -j REJECT
**that rule will also block the returning packages of an established connection**
***syn** blocks only packages for TCP connection initialization (important)

iptables -A FORWARD -p tcp -s 192.168.0.128/26 -d 192.168.0.64/26 \\
```

```
--syn -j REJECT

net0 -|<- net1 ( Router 1, FORWARD )

iptables -A FORWARD -p tcp -s 192.168.0.0/26 -d 192.168.0.64/26 \\
--syn -j REJECT

net0 -|<- net2 ( Router 2, FORWARD )

iptables -A FORWARD -p tcp -s 192.168.0.0/26 -d 192.168.0.128/26 \\
--syn -j REJECT

netExt -[80]->|- net0 (Router 0, FORWARD)

block outgoing DMZ traffic (syn)
iptables -A FORWARD -p tcp -s 192.168.0.0/26 -d 0/0 --syn -j REJECT

(only) allow web hosting in DMZ, e.g. for webmail etc.

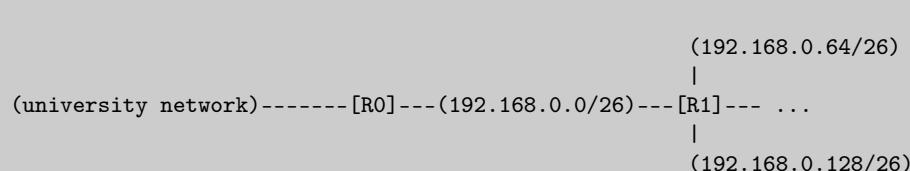
iptables -A FORWARD -p tcp -s 0/0 -d 192.168.0.0/26 --destination-port \\
80 --syn -j ACCEPT
iptables -A FORWARD -p tcp -s 0/0 -d 192.168.0.0/26 --syn -j REJECT
```

## 16.4 Refine Network Topology

Change topology to have one external and one internal Firewall/Router. The internal Firewall/Router has to have one interface per (sub)network faculty, department, work group, etc. For increasing number of internal networks, this will be realized by VLANs instead of having n-times NICs inside of the routing system established by switches that are able to create (separated) Virtual Local Area Networks.

Network Zones are as follows:

dmz (192.168.0.0/26) loc (192.168.0.64/26) loc2 (192.168.0.128/26)



Now, apply the firewall (IPTABLES) rules like described before.

## 16.5 Exercise: FW persistence

Iptables rules are not persistent by default and will be lost after rebooting. Also it is not very comfortable for the administrator to handle a huge set of rules. We have to make these iptables rules persistent and more manageable.

Possible solution: shorewall

```
apt-get install shorewall

/etc/default/shorewall

#Set shorewall to load when booting
startup = 1

/etc/shorewall/zones

Router 1:

#Zone      Description
r01        firewall
dmz        ipv4
loc        ipv4
loc2       ipv4

/etc/shorewall/interfaces
Router 1:

#Zone      interface      Broadcast      Options
dmz        eth0          detect         dhcp,nets=(192.168.0.0/26)
loc        eth1          detect         dhcp,nets=(192.168.0.64/26)
loc2       eth2          detect         dhcp,nets=(129.168.0.128/26)

/etc/shorewall/policy
Router 1:

#Src      Dest      Policy
dmz      all      DROP
loc      all      ACCEPT
loc2     loc      REJECT
loc2     all      ACCEPT
r01      all      ACCEPT
all      all      REJECT

/etc/shorewall/rules
Router 2:

#Action Source Dest Proto Dest.-Port
```

```
SECTION ESTABLISHED
```

```
SECTION RELATED
```

```
SECTION NEW
```

```
ACCEPT loc2    loc      tcp      80
ACCEPT loc2    loc      tcp      22
```

## 16.6 Transfer

Remove physical routing machines (R1 and R2) Bring all host to first switch (S0)( Class B Network with 16 (possible) subnetworks, e.g. one for each faculty( Install two minimalistic debian virtual machines( Enable the first virtual machine for routing (forwarding, 2 NICs) Set eth0 to IP of DMZ Zone Set eth1 to IP of locN Zone (N = 1-13)( Setup the second virtual machine as host within the locN Zone (N = 1-13) Setup FW of Router1 - Router13 to allow incoming ssh and www drop everything from dmz



In this case, R0 has to have all routing information to R1 until R13 In real world R1-R13 will be realized in one router with multiple interfaces, or (maybe more realistic) by creating VLANs



# Useful administration tools

---

## Contents

<b>17.1 cron - The Task Scheduling Daemon . . . . .</b>	<b>105</b>
17.1.1 What is cron? . . . . .	105
17.1.2 Different versions of cron . . . . .	107
17.1.3 run-parts . . . . .	109
<b>17.2 date - The System Date and Time . . . . .</b>	<b>109</b>
17.2.1 general syntax . . . . .	109
17.2.2 format sequences . . . . .	109
<b>17.3 find - search for files and directories in the file system . . . . .</b>	<b>110</b>
17.3.1 General syntax . . . . .	110
17.3.2 Advanced options . . . . .	110

---

*cron*, *date* and *find* are among the most useful tools for day to day work in a computer center. Because they can be used in a huge variety of ways, we have dedicated them an extra section.

## 17.1 cron - The Task Scheduling Daemon

### 17.1.1 What is cron?

Cron is a service program running as daemon. It is used to execute recurring tasks automatically at regular intervals, that can be defined very precisely. Automation can save a lot of time for a system administrator making his / her life much happier.

#### How to start cron? / Is cron running?

If the system provides good start-stop-script those should be used for easy tasks like the following.

To check whether cron is running the start-stop-script can be used:

```
$ /etc/init.d/cron status
```

Stop cron using the start-stop-script:

```
$ /etc/init.d/cron stop
```

To start cron using the start-stop-script:

```
$ /etc/init.d/cron start
```

### How to use cron? The crontab.

The crontab is the place (file) where is defined, what should be done, and when. You can consider this file a list of automated tasks or a table with one task in each row. Note that the last field is not mandatory. All text after a # is ignored.

#m	h	dom	mon	dow	user <sup>1</sup>	command	comment
5	*	*	*	*	flo <sup>1</sup>	/home/flo/test.sh	# test.sh will be executed every hour at 5 minutes past
16	7	*	*	1	root <sup>1</sup>	/home/flo/test_mon.sh	# test_mon.sh will be executed every monday at 7:16am

The first 5 fields define *when* and the last field defines *what* shall be done / executed. Each "when" field can contain values (see table), a \* that stand for all allowed values, a range of values (e.g. 1-3), a list of values (ex.: 2,4,7) or a combination of list and range (ex.: 1-3,6,9-11). It is also possible to specify fractions: every second, third, fourth and so forth value is \*/2, \*/3, \*/4.

Abbreviation	Write out	Range of allowed values	Explanation
m	minute	0-59	
h	hour	0-23	
dom	day of month	1-31	
mon	month	1-12 <sup>2</sup>	1 is January, 2 is February,...
dow	day of week	0-7 <sup>2</sup>	0 and 7 is Sunday, 1 is Monday, 2 is Tuesday,...

To create or edit the actual users crontab, use:

```
$ crontab -e # edit actual users crontab
```

This opens (or creates if not already existent) the crontab in the default editor of the user who executed the command.

<sup>1</sup>This field only exists in system-wide crontab or in crontabs stored in /etc/cron.d/. It defines which user's permissions the executed command will have.

<sup>2</sup>Names can also be used for the "month" and "day of week" fields. Use the first three letters of the particular day or month (case doesn't matter). Ranges or lists of names are not allowed.

The created files can be found in /var/spool/cron/crontabs/ but should only be edited by using the crontab -e command. If you have superuser (root) privileges you can edit crontabs of users as follows:

```
$ crontab -u <username> -e                                # edit <username>'s crontab
```

The -u <username> option can also be used in conjunction with the following options:

```
$ crontab -l                                # list actual crontab
```

```
$ crontab -r                                # remove the actual crontab
```

### Who is allowed to use cron?

In a Debian / Ubuntu system it is the default that every user can edit his/her own crontab. This can be restricted by creating the following files: If the file /etc/cron.allow exists you must be listed in it to use cron. If the file /etc/cron.deny exists you must *not* be listed in it to use cron.

### Files and Directories

/etc/crontab is the system-wide crontab. Here are the default directories defined for hourly, daily, weekly and monthly jobs.

```
/etc/cron.hourly/      # jobs are executed every hour using run-parts
/etc/cron.daily/       # jobs are executed every day using run-parts
/etc/cron.weekly/      # jobs are executed every week using run-parts
/etc/cron.monthly/     # jobs are executed every month using run-parts
```

The crontabs that contain the user field are stored in /etc/cron.d/.  
The user crontabs are stored in /var/spool/cron/crontabs/.

#### 17.1.2 Different versions of cron

##### vixie-cron

All the above refers to **cron** version 3.x. Originally coded by Paul Vixie in 1987 version 3 was released in 1993. It is still the default cron for Debian and Ubuntu. Version 4.1 was renamed to

ISC Cron and has enhanced security capabilities and works fine with selinux. It is the default in openSUSE Linux<sup>3</sup>.

### **anacron**

**anacron** is a computer program that performs periodic command scheduling which is traditionally done by cron, but without assuming that the system is running continuously. Thus, it can be used to control the execution of daily, weekly, and monthly jobs (or anything with a period of n days) on systems that don't run 24 hours a day.

Most Unix systems are set up to run "housekeeping chores" such as log-rotation, unused files deletion, indexing local files for the search engine, etc.

With cron, often these tasks are scheduled to be executed overnight or another low-usage time to avoid straining the system. If the system is turned off at the time a given task should have been run, it will not be executed for that iteration.

In contrast, anacron makes sure that these commands are run at the specified intervals as closely as machine uptime permits. It is a cron-complement (it requires cron) for laptop and home PC users for periodic tasks.

### **fcrontab**

**fcrontab** is a cron-like scheduler with extended capabilities. It implements most of Vixie Cron's functionalities. But contrary to Vixie Cron, fcrontab does not need your system to be up 7 days a week, 24 hours a day: it also works well with systems which are running neither all the time nor regularly.

Thus fcrontab aims to be a replacement for vixie-cron and anacron.

Fcron also includes a useful system of options, such as: run jobs one by one, run jobs at fcrontab's startup if they should have been run during system down time, a better management of the mailing of outputs, set a nice value for a job...

### **cronie**

It is a fork of the original vixie-cron and has security and configuration enhancements like the ability to use pam and SELinux. It is the default cron in Fedora Linux<sup>4</sup>.

---

<sup>3</sup><http://www.opensuse.org/>

<sup>4</sup><http://www.fedoraproject.org/>

### 17.1.3 run-parts

`run-parts` is a command that is often used in conjunction with cron. It runs all executables in the given directory

```
$ run-parts /path/to/directory
```

Executables that contain other characters than lower or upper case letters, digits, underscores and hyphens in their file names are ignored (if the `-lsbsysinit` is not given). This ensures that old files (ex. `.dpkg-old`) or backup files (like `.back`) are not executed.

## 17.2 date - The System Date and Time

The `date` command can set and print the system time. Here will only be described, how to print the actual date in different formats to the standard output.

### 17.2.1 general syntax

```
$ date [OPTION]... [+FORMAT]
```

### 17.2.2 format sequences

format sequences	description	example
<code>%A</code>	locale's full weekday name	Wednesday
<code>%d</code>	day of month	02
<code>%H</code>	hour (24 hour notation)	17
<code>%m</code>	month	08
<code>%M</code>	minute	15
<code>%y</code>	last two digits of the year	09
<code>%Y</code>	year	2009

The following example is a nice time format for filenames, as it will sort files alphabetically in chronological order:

```
$ date +%Y-%m-%d_%H:%M:%S  
2009-08-17_17:15:23
```

## 17.3 find - search for files and directories in the file system

### 17.3.1 General syntax

```
$ find <path> [<options>]
```

`find` always works recursively. If applied to directory `/etc`, all contained files and subdirectories are also listed (if not configured otherwise).

Some Examples:

```
find .          # lists all files and directories in the current working
                 # directory (CWD)
find . -type f # lists only files in the CWD
find . -type d # lists only directories in the CWD
find /          # lists all files and directories on this host
find /home/xyz # lists all files and directories in /home/xyz
find / -group users # lists all files and directories which belong to
                     # group "users"
```

### 17.3.2 Advanced options

#### -mtime -ctime -atime

- `-mtime`: modification time [24h]
- `-ctime`: time of last change of attributes<sup>5</sup> [24h]
- `-atime`: access time [24h]

Some examples:

```
find / -mtime -n # lists files that have been modified in the last n*24 hours
find / -mtime +n # lists files that have been modified more than n*24 hours
                 # ago (have not been modified in the last n*24 hours)
```

```
$ find . -mtime -1      # lists all files and directories of the working
                         # directory that have been modified in the last
                         # 24 hours
$ find /home -mtime +2  # lists all files and directories in /home that have
```

---

<sup>5</sup>attributes in this case are: permissions and ownership (user and group)

```
# been modified last time at least 48 hours ago (has
# not been modified in the last 48 hours)
$ find / -atime -3          # lists all files and directories that have been
                             # accessed in the last 36 hours
# lists all files and directories of the parent direc-
# tory that have been accessed last time at least 240
# hours [~10 days] ago (has not been accessed in the
# last 240 hours [~10days])
$ find .. -atime +10
```

**-perm <oktet mask>**

Finds only files with the given oktet mask of permissions.

**-maxdepth <n>**

Finds only files that are n directories deeper in the file system tree, relative to the given path (start directory).

**-xdev**

Finds only files on the same file system (same partition) as the given path (start directory).



# Secure Remote Administration Tools: ssh and scp

---

## Contents

<b>18.1 General overview . . . . .</b>	<b>113</b>
<b>18.2 sshd - Secure Shell Deamon . . . . .</b>	<b>114</b>
<b>18.3 ssh Protocol . . . . .</b>	<b>114</b>
<b>18.4 The application of ssh (Secure Shell) . . . . .</b>	<b>114</b>
18.4.1 Typical syntax for remote login . . . . .	114
18.4.2 Typical syntax for remote command execution . . . . .	115
18.4.3 Forward X11 Connections . . . . .	115
18.4.4 Piping through SSH . . . . .	115
18.4.5 Password-less SSH logins . . . . .	116
18.4.6 SSH Port Forwarding . . . . .	117
18.4.7 Other Useful Options . . . . .	119
<b>18.5 scp - Secure Copy . . . . .</b>	<b>119</b>
18.5.1 Options for scp AND cp . . . . .	120
18.5.2 Options only for scp . . . . .	120
<b>18.6 Infrastructure . . . . .</b>	<b>120</b>
<b>18.7 root login via ssh . . . . .</b>	<b>121</b>

---

Administrators often have to do work on remote computers. This is an introduction to two very useful command line tools that use encrypted data transfer and are considered to be safe for use in insecure networks like the internet.

**ssh** (secure shell) is a tool that lets you work on a remote computer as if you were sitting in front of it and typing to its shell. Or you can only execute single commands remotely. **scp** (secure copy) can copy files very similar to the normal **cp** (copy) command, but provides the possibility to copy files from one host to another.

## 18.1 General overview

This introduction will mainly deal with the application of the ssh and scp command provided by the **OpenSSH** project.

OpenSSH is an open source alternative for the proprietary Secure Shell software suite offered by SSH Communications Security. The OpenSSH suite comes with probably every Linux distribution and provides the client programs ssh and scp, the server program sshd and some other programs which handle the encryption keys. It can also be used to tunnel network connections.

Maybe it is a little confusing that the protocol ("language") the client and server use to communicate is also called ssh (ssh protocol).

## 18.2 sshd - Secure Shell Deamon

ssh and scp are client programs and useless, if they cannot connect to a corresponding server program. Thus on the remote host the sshd must be running and be accessible. **Port 22 is the default port** sshd listens on but it can be configured to listen on any port number and on any number of ports.

## 18.3 ssh Protocol

SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary. SSH is a protocol that can be used for many applications. Some applications not mentioned at other places in this document:

- in combination with SFTP, as a secure alternative to FTP file transfer
- in combination with rsync to backup, copy and mirror files efficiently and securely
- for port forwarding or tunneling a port (not to be confused with a VPN which routes packets between different networks or bridges two broadcast domains into one.)
- for using as a full-fledged encrypted VPN
- for browsing the web through an encrypted proxy connection with SSH clients that support the SOCKS protocol.

## 18.4 The application of ssh (Secure Shell)

(ssh is the secure substitution for programs like telnet or rsh.)

### 18.4.1 Typical syntax for remote login

```
$ ssh <username>@<remotehost>
```

<username> has to be substituted by the **login name** on the remotehost

<remotehost> has to be substituted either by the **host name** of the remote host, if it has a proper name resolution (DNS or entry in /etc/hosts), **or** by the **ip address** of the remote host

#### 18.4.2 Typical syntax for remote command execution

```
$ ssh <username>@<remotehost> <command>
```

Substitution as above and <command> must be substituted by the command that shall be executed. For example

```
$ ssh <username>@<remotehost> who
```

executes the "who" command which lists all users logged on to the remotehost:

```
flo      :0          2009-07-03 15:13
```

#### 18.4.3 Forward X11 Connections

To use a graphical program that runs on a remote host on the local machine, the -X option must be set. This can be done for running a single program:

```
$ ssh -X <username>@<remotehost> <graphical programs name>
```

example:

```
$ ssh <username>@<remotehost> xeyes
```

or for tunneling all graphical programs to the local machine

```
$ ssh -X <username>@<remotehost>
```

(Every window of a graphical program started on the remote host will appear on the local machine.)

#### 18.4.4 Piping through SSH

ssh can be used similarly to any other shell command with pipes (e.g. tar, gzip, grep etc.). This means, ssh can be used as a "pipeline" for streams to a remote host, which will be then tunneled via the encrypted ssh connection. The syntax is as follows:

```
<localcommand> | ssh user@remotehost "<remotecommand>"
```

This will log in via ssh to "remotehost", and pipe the standard output of "localcommand" to the standard input of "remotecommand". Some example for this would be:

```
echo "test output" | ssh user@remotehost "mkdir /tmp/test; cd /tmp/test; cat > output"
```

This will print the string "test output" into the file `/tmp/test/output` after the folder `/tmp/test` has been created. Note that the output of the local echo command will be piped to the first remote command which is opening the standard input for reading, i.e. the "mkdir" and "cd" commands will not be affected by the output stream of the local echo command.

An actual useful example would be the following:

```
tar -cp /etc -f - | gzip | ssh user@remotehost "cat > /tmp/etcbackup$(date +%F)"
```

This will create a tarball of the local `etc` folder, pipe it locally to gzip and pipe this output again via ssh to the "remotehost", where it is written with "cat" into a file with the name "etcbackup" plus the actual date.

#### 18.4.5 Password-less SSH logins

If frequent connections to a remote host are being established via ssh, it is often annoying to type a password each time. ssh allows for a "password-less" logging by the use of shared public keys, that has to be installed on the ssh server for each host that wishes to connect to this server. For this to work, a RSA key pair has to be created first. On the local host, run the following command:

```
ssh-keygen -t rsa
```

Do not give a passphrase for this key, as this would have to be typed in each time the key is being used. After that, a file in the user's home folder is created: `.ssh/id_rsa.pub` which has to be copied to the remote host. This can be done e.g. with piping it through ssh:

```
cat ~/.ssh/id_rsa.pub | ssh user@remotehost 'cat >> .ssh/authorized_keys'
```

Make sure that the `.ssh` folder exists on the remote host beforehand.

After that, the user can ssh into "remotehost" without giving a password.

This is also particularly useful for embedding ssh or scp connections in scripts, like in the above examples (section 18.4.4 on page 115).

#### 18.4.6 SSH Port Forwarding

SSH has the functionality of *forwarding* ports between the client and the server through the encrypted connection. This is working with the standard ssh implementations, no extra software is needed. This is also called "encapsulation" or "tunneling". There are *forward* and *backward* tunnels possible.

##### Forward Tunnel

The syntax is as follows:

```
ssh -L <local_port>:<host_to_forward_to>:<remote_port> user@sshserver
```

This will open a port (local\_port) on the ssh client host. All connections to this port will be forwarded (tunneled) via the ssh connection to the remote host (ssh daemon). As an example, we have a situation, where a web server on port 80 is running on a host (Host A) and a firewall is blocking connections from outside to this host and port, but allowing ssh connections on port 22. To access the web server from the outside (Host B), an ssh connection with a port forwarding can be established:

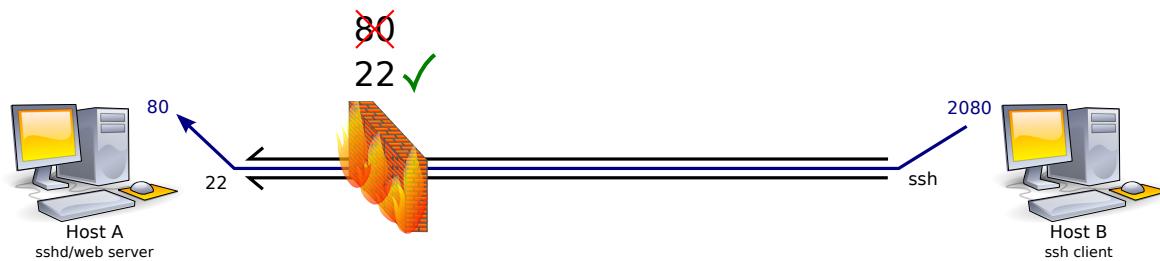


Figure 18.1: ssh forward tunnel

```
ssh -L 2080:localhost:80 user@sshserver
```

As long as this ssh connection remains established, all connections to the port 2080 on the client side (where ssh has been started, Host B) will be forwarded to the port 80 on the "sshserver" (Host A) host. In this case, one can type `http://localhost:2080` into a web browser on the client (Host B) and be directed to the web server (Host A) via the encrypted ssh connection.

Another scenario is an encrypted connection to services that are unencrypted, or when the connection is restricted to the localhost only (e.g. web interfaces from services like CUPS etc.).

### Reverse Tunnel

The syntax for a reverse tunnel is:

```
ssh -R <remote_port>:<host_to_forward_to>:<local_port> user@sshserver
```

As with the forward tunnel, a port will be opened for listening, but here on the remote host (ssh server). All connections to this port will be forwarded back to the ssh client host.

### Combined Forward and Reverse SSH Tunnel

Another scenario consists of two hosts (Host A and Host B) which each are behind NAT firewalls and thus cannot access each other directly. A third host (Host C) on the Internet which runs an ssh server can act as an intermediate host:

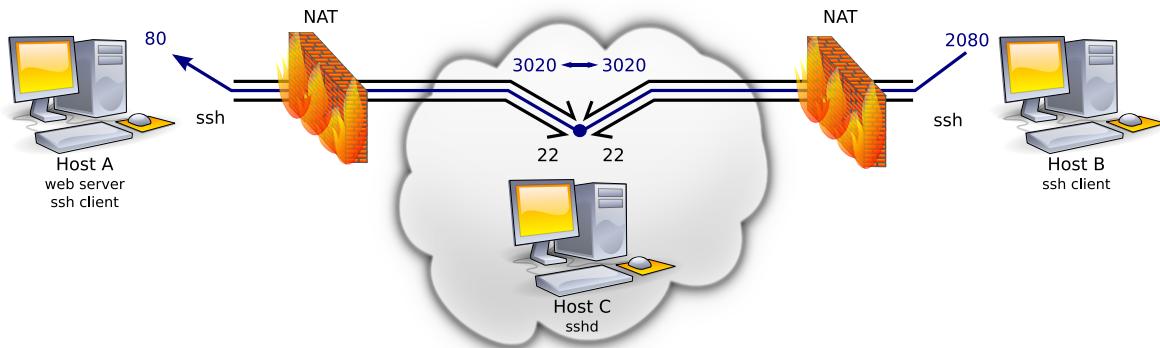


Figure 18.2: combined forward and reverse tunnel

In this example, Host B establishes a forward SSH tunnel to Host C:

```
ssh -L 2080:localhost:3020 user@Host_C
```

Host A (with e.g. a web server on port 80) establishes a reverse SSH tunnel to Host C:

```
ssh -R 3020:localhost:80 user@Host_C
```

As long as both ssh connections remain established, it is possible to access the web server on Host A by pointing a web browser on Host B to <http://localhost:2080>. All packets to port 2080 on Host B will be tunneled to Host C on port 3020. This matches the remote port of the reverse tunnel which again forwards the packets to Host A on port 80.

Similarly, it is possible to concatenate forward tunnels, i.e. by connecting from a client to one host via ssh and from there, connect to another host with a new ssh tunnel; the remote port of the first tunnel has to match the source port of the second tunnel.

#### 18.4.7 Other Useful Options

-p <portnumber> : specify a different than the default port number

see:

```
$ man ssh
```

## 18.5 scp - Secure Copy

(scp is the secure substitution for programs like rcp.)

The base syntax is very similar to syntax of the cp command:

```
$ cp <source> <destination>
$ scp <source> <destination>
```

Both <source> and <destination> can be substituted either by a relative or an absolute path for both commands. Unlike cp with scp <source> and <destination> can be substituted by <username>@<remotehost>:<path>

Examples:

```
$ scp <username>@<remotehost>:<sourcepath> <destinationpath>
```

copies a file from a remote host to the local machine

```
$ scp <sourcepath> <username>@<remotehost>:<destinationpath>
```

copies a file from the local machine to a remote host

```
$ scp <username>@<remotehost1>:<sourcepath> <username>@<remotehost2>:<destinationpath>
```

copies a file from remote host 1 to remote host 2

### 18.5.1 Options for scp AND cp

-r : copy recursively

-p : preserve owner, group, permissions and times

### 18.5.2 Options only for scp

-P <portnumber> : specify a different than the default port number (beware: ssh uses a lower case “p” where scp uses a upper case ‘P’)

see:

```
$ man scp
```

## 18.6 Infrastructure

Installation (on debian / ubuntu):

```
$ aptitude install openssh-server
```

Configuration files for ssh and sshd:

```
/etc/ssh/
```

User configuration files for ssh:

```
~/.ssh/
```

File that stores public key fingerprints of all SSH servers contacted:

```
~/.ssh/known_hosts
```

## 18.7 root login via ssh

The all powerful user “root” should not be allowed to directly log into a server via ssh. This makes it more difficult to attack a server.

To disable direct root logins, configure in `/etc/ssh/sshd_config`:

```
PermitRootLogin no
```



# Partitions, LVM and RAID

---

## Contents

<b>19.1 File systems . . . . .</b>	<b>124</b>
<b>19.2 Partitions . . . . .</b>	<b>124</b>
<b>19.3 In Practice: Add a partition to the file system tree . . . . .</b>	<b>125</b>
19.3.1 Preliminary considerations . . . . .	125
19.3.2 Check the System . . . . .	125
19.3.3 Create a partition . . . . .	126
19.3.4 Format the new partition . . . . .	127
19.3.5 mount - Integration of File Systems to the File System Tree . . . . .	127
19.3.6 cp -a - Copy Data . . . . .	127
19.3.7 fstab - Fix Position in File System Tree . . . . .	128
19.3.8 Testing and Finishing . . . . .	128
<b>19.4 Logical Volume Manager . . . . .</b>	<b>129</b>
19.4.1 Installation of LVM . . . . .	129
19.4.2 LVM configuration . . . . .	129
<b>19.5 RAID . . . . .</b>	<b>131</b>
19.5.1 RAID levels . . . . .	132
19.5.2 Hardware vs. Software RAID . . . . .	134
19.5.3 Linux Software RAID . . . . .	134

---

This chapter gives an overview about file systems and partitioning of hard drives. The process of adding a partition to a file system tree is explained step by step.

This common scheme of storage media usage is rather old and lacks both flexibility and stability for today's scenarios in servers and workstations. Partitions cannot be modified, moved or extended in an easy and comfortable way. Also, hard drive failure is a common threat which can result in downtimes.

The Linux "Logical Volume Manager" is a layer between the physical partitions and the file systems, which allows for easy grouping, re-grouping, resizing, snapshotting etc. of volumes, even without interrupting normal operation.

RAID technologies increase the reliability and stability of storage media by distributing the data over several physical storage media, thus creating redundant and transparent storage devices.

## 19.1 File systems

A file system defines the way in which files are named and where they are logically placed for storage and retrieval. In Unix-like OSes, Windows as well as MacOS and others, files are placed in a hierarchical (tree) structure, in directories or subdirectories.

File systems specify naming conventions for files, including the maximum number of characters a file name can have, which characters are allowed, the length of the suffix etc. Also, the file system defines the format for the path to a file through the directory structure.

There are many different file systems supported in Linux. To name only a few that are designed for Linux with different purposes: Ext2, Ext3, Ext4, ReiserFS, XFS, JFS. Some of those file systems use so-called *journaling* file systems. These file systems are logging all changes to a journal before committing them. This increases robustness, as in case of a power failure or system crashes journaling file systems are less likely to become corrupted.

Linux as well the Windows support file the systems FAT and NTFS, HFS+ from MacOS/X as well as the file systems of other Unix-like operating systems like Solaris.

The Linux file system root (highest point in the hierarchy) is denoted by `/`.

## 19.2 Partitions

Partitions are logical divisions of hard disks or solid state disks, that are created to get the appearance of having separate drives, for different operating systems or different purposes. One distinguishes primary partitions, extended and logical ones.

For both Windows and Linux on PC-compatible hardware the partition information or partition table is stored in the master boot record (MBR) of the respective hard disk. On other systems and architecture the partition is stored in other tables or disk labels, we are not referring to these here.

A PC hard disk is divided into at most four, at least one primary partitions. One of those four can also be an extended partition which can contain *secondary* or *logical* partitions. The primary partitions are described by 16-bit entries in the *partition table*, containing the beginning of the partition, the end and its type.

In the following, we want to describe the practical procedure of adding a new partition and mounting it to the file system tree.

## 19.3 In Practice: Add a partition to the file system tree

### 19.3.1 Preliminary considerations

It is not always possible to add a new partition to the hard disk. If the system lacks hard disk space first consider the following options:

- delete unnecessary data, avoid redundant data
- compress data
- increase size of an existing partition

In addition one has to notice the following facts:

- A partition can only be unmounted if it is not in use (i.e. no processes are using any files on it).
- It is not possible to swap out every directory from the root (/) partition. The programs that are necessary for the system boot have to stay on the root (/) partition: `/bin /etc /lib /root /sbin`
- On large systems it is common to have separated partitions for `/usr`, `/home` and/or `/var`. Here we practice how to move the `/home` directory onto a separate partition.
- Most operations need root privileges: Make sure you are working as root and log out after finishing work.
- Always use these commands with caution, as they can easily destroy all data on the hard drive.

### 19.3.2 Check the System

How does our file system tree looks like? What is currently mounted<sup>1</sup>?

```
$ mount
/dev/sda2 on / type reiserfs (rw,acl)          ##### <-- real partition
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,nosuid,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
```

---

<sup>1</sup>Mounting is the process of making a file system ready for use by the operating system, i.e. reading certain index data from the storage into the memory.

```

lrm on /lib/modules/2.6.34-13-generic/volatile type tmpfs (rw,mode=755)
/dev/sda1 on /boot type ext3 (rw,relatime)      ##### <-- real partition
securityfs on /sys/kernel/security type securityfs (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,
nodev)

```

How many space is in use for the data in */home*:

```
$ du -hs /home/
3.5G   /home/
```

Home has a size of 3.5GB so we need a partition larger than that. The following command provides an overview over all available hard disks:

```

$ fdisk -l

Disk /dev/sda: 160.0 GB, 160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x50bf8b46

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           1         33     265041   83  Linux
/dev/sda2            34       1308    10240800   82  Linux
/dev/sda3            1309      1830    4192704+   82  Linux swap / Solaris
/dev/sda4            1831      19457   141588877+   5  Extended
```

Conclusion: We have one hard disk with an overall size of 160GB. The first three partitions (*sda1* to *sda3*) are used. *sda1* is used as */boot* and *sda2* is used as */*. *sda3* is probably used as swap partition (to be sure use *swapon -s* respective *swapon /dev/sda3*). There is an extended partition which contains no logical partitions but has lots of free space.

### 19.3.3 Create a partition

There are several tools for the creation of partitions for the command line as well as graphical ones: as command line tools we propose *fdisk*, we used before to list the existing partitions or *cfdisk*, which is easier to use than *fdisk*. Most Linux distribution ship with a graphical partitioning tool like *gparted*.

```
$ cfdisk /dev/sda
```

Move down to the "free space" and select "new" to create a new partition. Then, select "logical" and the size of the new partition. If everything is correct select "write" and "quit".

```
/dev/sda5
```

(*/dev/sda4* is the device file for the extended partition)

#### 19.3.4 Format the new partition

```
$ mkfs.ext4 /dev/sda5
```

Alternatively other different file systems (like ReiserFS, XFS, ...) can be used. Once formatted the partition can be integrated to the file system tree.

#### 19.3.5 mount - Integration of File Systems to the File System Tree

To connect a file system (on a partition of a hard disk or a usb-stick or any other media) to the file system tree often a directory under */mnt* like */mnt/tmp* or the */mnt* directory itself is used. Actually any directory can be used. To connect the file system on */dev/sda5* to the file system tree in */mnt/tmp* use:

```
$ mount /dev/sda5 /mnt/tmp
```

To be sure that */dev/sda5* "is mounted" in */mnt/tmp* just type:

```
$ mount  
/dev/sda2 on / type reiserfs (rw,acl)  
[...]  
/dev/sda1 on /boot type ext3 (rw,relatime)  
/dev/sda5 on /mnt/tmp type ext3 (rw,relatime)
```

#### 19.3.6 cp -a - Copy Data

Not to change any data, it is necessary to use **cp** with the **-a** (archive) option. This preserves all permissions, owners, groups and time stamps.

```
$ cp -a /home/* /mnt/tmp
```

To get an brief overview if all users own their home directory:

```
$ ls -l /mnt/tmp
```

Now we can disconnect the new partition

```
$ umount /mnt/tmp
```

and mount it to its supposed place.

### 19.3.7 fstab - Fix Position in File System Tree

fstab stands for "file system **table**" and contains static file system information:

```
$ cat /etc/fstab
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/sda2      /
/dev/sda1      /boot    ext3   defaults,relatime 0   1
/dev/sda3      none     swap   sw             0   0
[...]
```

We need to add a line for the new home partition. Use your preferred editor and add the following line to */etc/fstab*:

```
/dev/sda5      /home     ext3   defaults,relatime 0   2
```

If we now type:

```
$ mount /home
```

The mount command checks the */etc/fstab* and finds the appropriate entry and connects */dev/sda5* to */home*

### 19.3.8 Testing and Finishing

In this configuration the system shall work normally without the user noticing any changes. For security, just double check: Has all data been copied? Do all files and folders have the right owners, groups, permissions and time stamps? Can users log normally in again?

Until now the old data still exists on the root `(/)(/dev/sda2)` partition, but it is overridden by the data of the new home partition `(/dev/sda5)`. To free the space on `sda2`, it is necessary to umount `/dev/sda5`. Check whether it's really unmounted and remove the contents of `/home`. After that, re-mount `/dev/sda5` again:

```
$ umount /home
$ mount
/dev/sda2 on / type reiserfs (rw,acl)
[...] # /dev/sda5 should not appear
/dev/sda1 on /boot type ext3 (rw,relatime)
$ rm -r /home/*
$ mount /home
```

Congratulations you just added a partition to the file system tree.

## 19.4 Logical Volume Manager

To use LVM under Linux, first physical partitions have to be added to a "pool" of partitions for use with LVM. This is done with the `pvcreate` command. Physical volumes can then be grouped into "volume groups" with the command `vgcreate` (this represents "virtual" hard drives). Finally inside volume groups, "logical volumes" can be created with the `lvcreate` command, you can refer to this as "virtual partitions".

### 19.4.1 Installation of LVM

Under Ubuntu, the needed packages can be installed easily:

```
$ sudo apt-get install lvm2 lvm-common
```

If not done automatically, the newly installed modules need to be loaded:

```
$ modprobe dm-mod
```

### 19.4.2 LVM configuration

The physical partitions on the hard drive need to be set to type "Linux LVM" (8e). To change the partition type, type `t` at the `fdisk` prompt.

It is best practice to divide the hard drive into several smaller physical partitions of a fixed size. These can be grouped into logical volumes; later, these physical partitions can be added to the existing logical volumes as needed. This allows for a fair amount of flexibility.

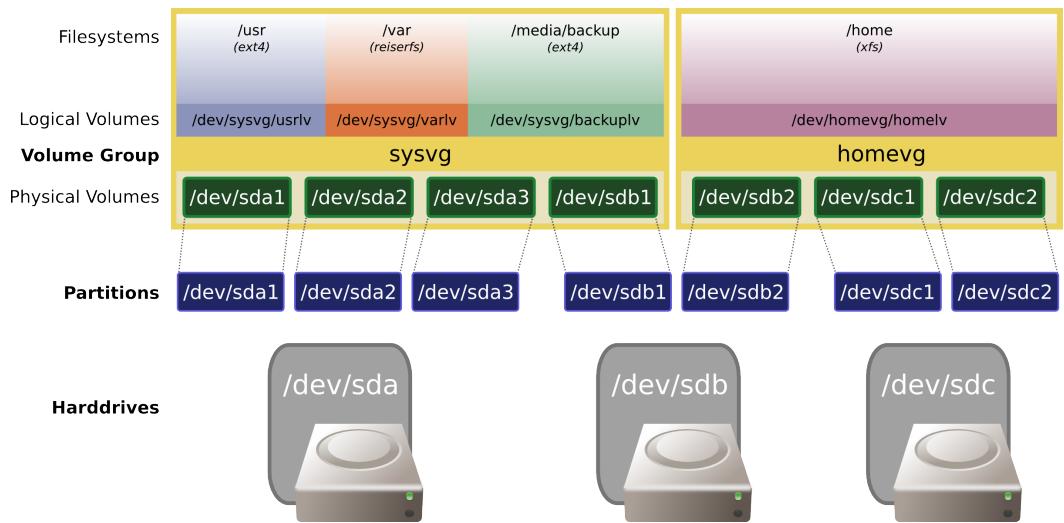


Figure 19.1: Example LVM configuration

As a practical example, the following scenario will be created:

Three hard drives (sda, sdb, sdc) have been divided into seven partitions and are added as "physical volumes" into the LVM pool:

```
$ pvcreate /dev/sda1 /dev/sda2 /dev/sda3 /dev/sdb1 /dev/sdb2 /dev/sdc1 /dev/sdc2
```

The command `pvscan` prints all registered physical volumes in LVM.

After that, the physical volumes can be grouped into two volume groups "sysvg" and "homevg":

```
$ vgcreate sysvg /dev/sda1 /dev/sda2 /dev/sda3 /dev/sdb1 $ vgcreate homevg /dev/sdb2 /dev/sdc1 /dev/sdc2
```

The command `vgscan` prints all existing volume groups on the system. `vgdisplay` prints more information about them, e.g. size in the unit of "PE". This stands for (virtual) "physical extents" corresponding to "blocks" on physical devices. The size of one PE is stated in the output of `vgdisplay`.

Inside the volume groups, logical volumes can then be created. The `-l` parameter defines the size of the volume in PE. The `-n` parameter specifies the name of the new logical volume.

```
$ lvcreate -l 9540 sysvg -n usrlv $ lvcreate -l 9540 sysvg -n varlv
$ lvcreate -l 13540 sysvg -n backuplv $ lvcreate -l 30000 homevg -n homelv
```

After these commands, the volumes as shown in the diagram are available and ready to be used. They are accessible as device files in the `/dev` folder under `/dev/<vgname>/<lvname>`. Finally, filesystems can be created:

```
$ mkfs.ext4 /dev/sysvg/usrlv $ mkfs.reiserfs /dev/sysvg/varlv $ mkfs.ext4 /dev/sysvg/backuplv  
$ mkfs.xfs /dev/homevg/homelv
```

The procedure for adding another physical volume later to a volume group is with the `vgextend` command. Please be aware that you *should* unmount your volume before doing this. Some file systems like ext3 support online resizing (i.e. while being mounted) which is considered "safe", but you never know :)

First, the new physical partition (sdd1, as an example) has to be added again as physical volume:

```
$ pvcreate /dev/sdd1
```

Then, this physical volume can be added to an existing volume group:

```
$ vgextend homevg /dev/sdd1
```

This adds space to the volume group which allows the size of a logical volume to be increased. For the amount of free PE, use the `vgdisplay` command again. Then, resize the logical volume with e.g.:

```
$ lvresize -l 40000 /dev/homevg/homelv
```

Existing file systems on the logical volume need to be resized, too. The procedure for this depends on which file system is being used. E.g. for ext2/ext3 file systems, use the command `resize2fs`, for reiserfs, use `resize_reiserfs`

This has been just a brief overview of the basic functions of LVM. Many more commands and options for managing LVM are available. Please check the man pages or the official LVM wiki pages: <http://www.tldp.org/HOWTO/LVM-HOWTO/index.html>

## 19.5 RAID

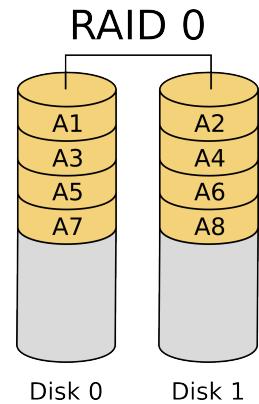
RAID is a technology used to combine multiple storage media into arrays to divide and replicate data among them and thus increasing fault tolerance and/or read-write performance. RAID stands for **R**edundant **A**rray of **I**ndependent **D**isks. When multiple physical disks are set up to use RAID technology, they are said to be in a RAID array. This array distributes data across multiple disks, but the array is addressed by the operating system as one single disk, so the user or the applications will not take notice of this. RAID can be set up to serve several different purposes referred to as RAID "levels", spanning from RAID 0 to RAID 6<sup>2</sup>. The most common RAID levels will be briefly presented here.

<sup>2</sup>More exotic variants and combinations of these levels exist but will not be covered in this documentation

### 19.5.1 RAID levels

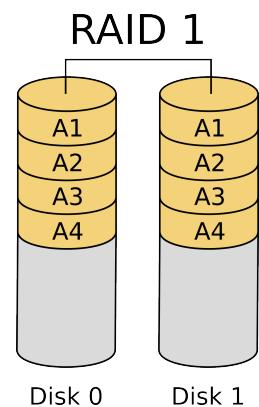
#### RAID 0

RAID 0 (block-level striping without parity or mirroring) provides improved performance and additional storage but no redundancy or fault tolerance (making it not true RAID, according to the acronym's definition). Any disk failure destroys the array, and the likelihood of failure increases with more disks in the array. A single disk failure destroys the entire array because when data is written to a RAID 0 volume, the data is broken into fragments called blocks. The blocks are written to their respective disks simultaneously on the same sector. This allows smaller sections of the entire chunk of data to be read off the drive in parallel, increasing bandwidth. RAID 0 does not implement error checking, so any error is uncorrectable. More disks in the array means higher bandwidth, but greater risk of data loss.



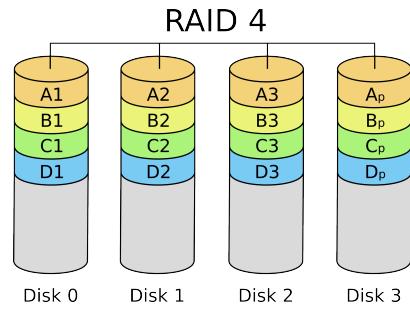
#### RAID 1

In RAID 1 (mirroring without parity or striping), data is written identically to multiple disks (a "mirrored set"). Although many implementations create sets of 2 disks, sets may contain 3 or more disks. This array level provides fault tolerance from disk errors or failures and continues to operate as long as at least one drive in the mirrored set is functioning. Increased read performance occurs when using a multi-threaded operating system that supports split seeks, as well as a very small performance reduction when writing.



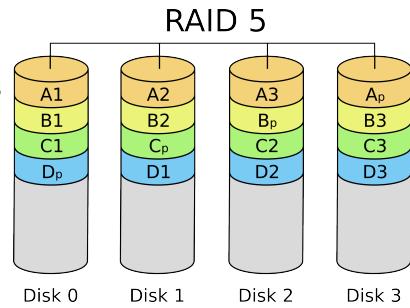
### RAID4

RAID4 (block-level striping with dedicated parity) is identical to RAID 5 (see below), but writes all parity data to a single disk, which can create a performance bottleneck. In this setup, files can be distributed between multiple disks, as the data to be written is divided into small chunks ("blocks") – the first block is written to the first disk, the second to the next, and so on. Each disk operates independently which allows I/O requests to be performed in parallel, though data transfer speeds can suffer due to the type of parity. The error detection is achieved through dedicated parity and is stored in a separate, single disk unit.



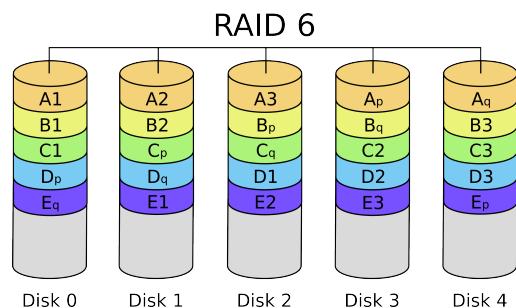
### RAID5

RAID 5 (block-level striping with distributed parity) distributes parity along with the data and requires all drives but one to be present to operate; drive failure requires replacement, but the array is not destroyed by a single drive failure. If a drive fails, all data can be calculated from the distributed parity on the remaining disks, so the array will continue to operate as normal. If a second drive will fail, the array will lose all data, though. Therefore, failed drives need to be replaced as soon as possible. After replacing a failed drive, the array stays vulnerable until the new drive has been completely rebuilt from the data and the parity information on the remaining drives.



### RAID6

RAID 6 (block-level striping with double distributed parity) provides fault tolerance from two drive failures; each parity information is saved twice and thus the array continues to operate with up to two failed drives. This makes larger RAID groups more practical, especially for high-availability systems. This becomes increasingly important as large-capacity drives lengthen the time needed to recover from the failure of a single drive. Single-parity RAID levels are as vulnerable to data loss as a RAID 0 array until the failed drive is replaced and its data rebuilt; the larger the drive, the longer the rebuild will take. Double parity gives time to rebuild the array without the data being at risk if a single additional drive fails before the rebuild is complete.



## RAID Overview

RAID Specifications overview:

Level	Description	Minimum no. of disks	Space Efficiency	Fault Tolerance	Read Benefit	Write Benefit
RAID 0	Block-level striping without parity or mirroring	2	1	0 (none)	nX	nX
RAID 1	Mirroring without parity or striping	2	1/n	n-1 disks	nX	1X
RAID 4	Block-level striping with dedicated parity	3	1-1/n	1 disk		
RAID 5	Block-level striping with distributed parity	3	1-1/n	1 disk		
RAID 6	Block-level striping with double distributed parity	4	1-2/n	2 disks		

### 19.5.2 Hardware vs. Software RAID

RAID systems can be implemented in hardware (on a special RAID controller) or in software (by a driver of the operating system). Each implementations have advantages and disadvantages.

Software-based RAID typically creates more system load, because the CPU has to calculate all the RAID operations, the more drives are included in the array (parity calculations, distributing the read/write operations) and the bus to the hard drives has to carry more data for writing. With modern CPUs however, this decrease is insignificant on most systems. Also, sometimes, booting from a software RAID is difficult or impossible due to boot loaders not supporting RAID arrays. As most hardware-based RAID controllers are using proprietary formats and are not interchangeable, software-based RAID has the advantage of not being dependent on a specific manufacturer and (if, for example, the controller dies) can be always restored on a different machine running the same operating system.

Hardware RAID controllers use different, proprietary disk layouts, so it is not usually possible to span controllers from different manufacturers. They do not require processor resources, the BIOS can boot from them, and tighter integration with the device driver may offer better error handling. Most hardware implementations provide a read/write cache, which, depending on the I/O workload, will improve performance. In most systems the write cache is non-volatile (i.e. battery-protected), so data loss does not occur on a power failure during write operations.

### 19.5.3 Linux Software RAID

Linux supports software-based RAID in the levels RAID 0, RAID 1, RAID 4, RAID 5, RAID 6 and all layerings of the above (i.e. combinations of different RAID levels). In the following example, two hard disks (sda and sdb) which are of exactly the same size, will be divided into two partitions each, a small one to carry swap space and a larger one carrying user data. The corresponding partitions of the two drives will be put into RAID 1 arrays.

## Configuration

Array	Drive 1	Drive 2	size
	sda	sdb	500 GB
md1	sda1	sdb1	10 GB
md2	sda2	sdb2	490 GB

The hard drives need to be partitioned using e.g. fdisk as explained above. The sizes of the partitions to be put into an array have to have the exact same size (number of blocks). The type of all partitions to be put into a RAID array need to be "Linux raid autodetect" (fd).

The main command for setting up and maintaining software RAIDs under Linux is `mdadm`. For a comprehensive list of all parameters and options, see the man page.

To setup initialize to two arrays, simply type:

```
$ mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sda1 /dev/sdb1
$ mdadm --create /dev/md2 --level=1 --raid-devices=2 /dev/sda2 /dev/sdb2
```

This creates two new device files, `/dev/md1` and `/dev/md2`, which make the arrays accessible to the system. The `--level` option specifies the RAID level. After this, the arrays can be formatted just as normal drives. According to our example:

```
$ mkswap /dev/md1
$ mkfs.ext4 /dev/md2
```

The configuration of the arrays will be stored automatically by `mdadm` in `/etc/mdadm/mdadm.conf` and re-read at boot time. Should it fail to do so, this will bring an array back:

```
$ mdadm --assemble /dev/md1 /dev/sda1 /dev/sdb1
$ mdadm --assemble /dev/md2 /dev/sda2 /dev/sdb2
```

Do *not* use the `--create` option in this case, because this would create a new array and thus wiping all data of the old one. Alternatively, `mdadm --assemble --scan` will assemble any RAID arrays it can detect automatically.

The file `/proc/mdstat` dynamically shows information about all RAID arrays on the system. If e.g. a drive of an array fails, it can be seen here:

```
$ cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sda1[0] sdb1[1]
```

```
488375044 blocks [2/2] [UU]
md2 : active raid1 sda2[0] sdb2[1]
      983750 blocks [2/2] [UU]
```

Most important part is the two [UU], which indicate, that both drives are up and running. If one drive fails, it would print e.g. [U\_].

### Replacing failed hard drives

When this happens, the drive can be replaced while the array is running. To do this, first the drive has to be marked as "failed" and then "removed" from the array, e.g.:

```
$ mdadm --manage /dev/md1 --fail /dev/sdb1
$ mdadm --manage /dev/md2 --fail /dev/sdb2
$ mdadm --manage /dev/md1 --remove /dev/sdb1
$ mdadm --manage /dev/md2 --remove /dev/sdb2
```

The failed drive can then be physically removed from the computer. Note that this is not always possible on all systems, especially on normal PCs, it often is not working and maybe even risking damage, so better shutdown the PC first. On most servers hardware, it should be no problem, though.

After inserting the replacement hard drive, the partitions have to be created on it according to the other drive(s) of the array. This can either be done manually using `fdisk` or with the `sfdisk` command, which can "copy" partition tables from one drive to another:

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

After that, the partitions of the new drive can be added to the array:

```
$ mdadm --manage \dev\md1 --add /dev/sdb1
$ mdadm --manage \dev\md2 --add /dev/sdb2
```

Now both arrays (`/dev/md1` and `/dev/md2`) will be synchronized. During this time, `cat /proc/mdstat` will look like this:

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid5] [raid4] [raid6] [
      raid10]
md1 : active raid1 sda1[0] sdb1[1]
      24418688 blocks [2/1] [U_]
      [=]>..... recovery = 9.9% (2423168/488375044) finish=2.8
           min speed=127535K/sec

md2 : active raid1 sda2[0] sdb2[1]
      24418688 blocks [2/1] [U_]
      [=]>..... recovery = 6.4% (1572096/983750) finish=1.9min
           speed=196512K/sec
```



# LAMP - Linux Apache MySQL PHP

---

## Contents

<b>20.1 Introduction</b>	139
<b>20.2 Apache HTTP Server</b>	140
20.2.1 Installation	140
<b>20.3 MySQL</b>	142
20.3.1 Installation	142
20.3.2 Administrering via CLI	143
20.3.3 Administering via phpMyAdmin	144
<b>20.4 PHP</b>	146
20.4.1 Installation	146
20.4.2 A Server-side Scripting Language	146
<b>20.5 Setting up Web-based open source software with LAMP</b>	147
20.5.1 A internal Knowledge Management System	147
20.5.2 A Personal Publishing Platform	151
20.5.3 A popular Internet Forum Package	152

---

This chapter gives a brief introduction to a “LAMP” system, what it is and how it can be set up.

## 20.1 Introduction

LAMP is an acronym. It stands for a special web server system including **L**inux as operating system, **A**pache as web server, **M**ySQL as database system and **P**HP (seldom also Python or Perl) as server-side scripting language. Thus, PHP is most common, this chapter will only refer to this programming language. LAMP is a very popular web application software stack to serve dynamic web pages (Figure 20.1 shows the environment of web-based software development with LAMP).

Similar terms exist for essentially the same software suite (AMP) running on other operating systems, such as MS Windows (WAMP), Mac OS (MAMP), Solaris (SAMP), or OpenBSD (OpAMP).

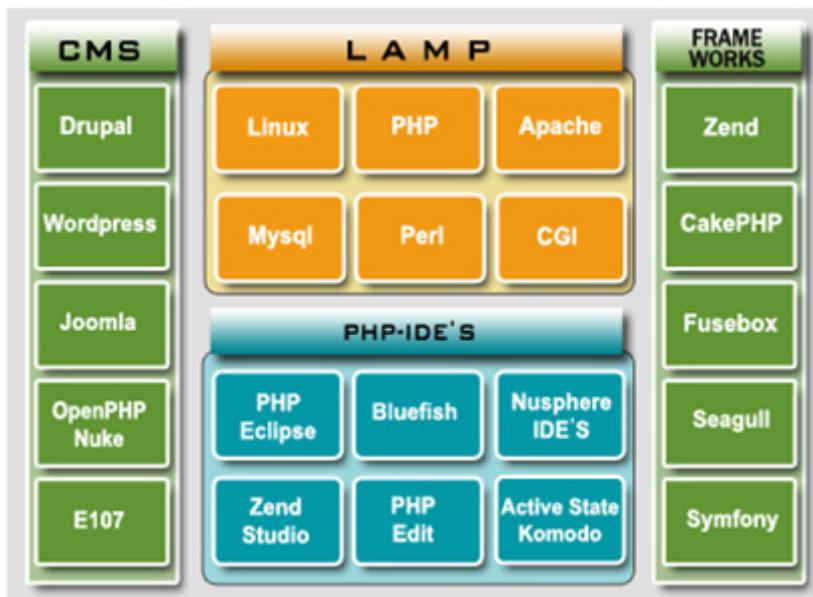


Figure 20.1: programming models for developing web-based software with LAMP

## 20.2 Apache HTTP Server

Apache is a popular web server usually run on Linux operating system. It is highly modularized and supports a lot of features (like virtual hosting and encrypted connections), is scalable, stable and released as open source, which makes it the most popular web server on the Internet ( 60% of all web pages worldwide are running on Apache).

### 20.2.1 Installation

To install it on a Debian/Ubuntu system, simply type

```
$ sudo aptitude install apache2
```

To test the Apache server, enter the following commands

```
$ apache2 -v                                # check the version
$ sudo /etc/init.d/apache2 stop                # stop the service
$ sudo /etc/init.d/apache2 start               # start the service
$ sudo /etc/init.d/apache2 restart            # restart the service
$ sudo /etc/init.d/apache2 reload             # reload the configuration of Apache
```

Now we have a running web server :-)

If you get a message like this:

```
apache2: Could not reliably determine the server's fully qualified domain name,
          using 127.0.0.1 for ServerName
```

it probably means, that your hostname was not set properly. For practicing this doesn't matter, but if you want to correct it and get rid of the error message, you need a *fully qualified domain name* (FQDN) in the form of:

```
<host>.<domain>
```

To correct this, edit `/etc/hostname` to

```
<yourhostname>
```

and add an entry to `/etc/hosts`

```
127.0.0.1 <yourhostname>.<yourdomainname> <yourhostname>
```

Check by running

```
$ hostname -s
<yourhostname>
$ hostname -f
<yourhostname>.<yourdomainname>
```

If you corrected the FQDN you can restart Apache without the complaint:

```
$ sudo /etc/init.d/apache2 restart
 * Restarting web server apache2               [ OK ]
```

You can point your web browser (ex. Firefox) to your machine (enter as URL `<yourhostname>`) now, and you will see a page that tells you "It works!"

## 20.3 MySQL

MySQL is a relational database management system that has more than 6 million installations worldwide. MySQL stands for **M**y **S**tructured **Q**uery **L**anguage. The program runs as a server providing multi-user access to a number of databases.

The project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements (“dual licensing”).

MySQL is often used in free software projects that require a full-featured database management system, such as WordPress, phpBB and other software built upon the LAMP software stack. It is also used in very high-scale World Wide Web products including Wikipedia.

One big advantage is that it runs on many different platforms.

### 20.3.1 Installation

```
$ sudo aptitude install mysql-server
```

When asked for the New password for the MySQL “root” user supply a long and difficult password, as data security relies on it! If you have not been asked to set a root-password for MySQL, you can do this by running:

```
/usr/bin/mysqladmin -u root password <yourDifficultPassword>
```

Let’s take a look at the open network ports:

```
$ lsof -i -P | grep mysqld
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME [...]
mysqld 2310 mysql 11u IPv4 1807383      TCP localhost:3306(LISTEN) [...]
```

The first line (beginning with the process name *mysqld*) shows the MySQL server listening on a localhost interface on port 3306.

MySQL also listens for connections on a file */var/run/mysqld/mysqld.sock*. This file is a *socket*, that can be used similar to a network connection.

MySQL installation done :-)

### 20.3.2 Administratering via CLI

#### mysql

is a simple SQL shell (with GNU readline capabilities). It supports interactive and non interactive use. When used interactively, query results are presented in an ASCII-table format.

The following example will show you how to create a database via CLI (command Line Interface), if you want to set up a web-based application in a LAMP environment.

At first you should log into the system

```
$ mysql -u root -p  
# if the password is not set to user root  
$ mysql -u root
```

To create a new database with name "myDB" via CLI

```
mysql> create database myDB;  
Query OK, 1 row affected (0.03 sec)  
  
mysql>
```

To delete the databases from MySQL server via CLI

```
mysql> show databases;          # display all served databases on the server  
mysql> drop database myDB;    # delete the database "myDB"  
mysql> exit                   # quit
```

#### mysqladmin

is a client for performing administrative operations. You can use it to check the server's configuration and current status, to create and drop databases, and more.

To create or delete a new database from MySQL server

```
$ mysqladmin -u root -p create myDB      # create a new DB "myDB"  
$ mysqladmin -u root -p drop myDB        # delete a DB "myDB"
```

To create the first new password for user **root**

```
$ mysqladmin -u root password YOUR_PASSWD
```

To change the old password for user `root`

```
$ mysqladmin -u root -p old-password YOUR_NEW_PASSWD
```

### 20.3.3 Administering via phpMyAdmin

PhpMyAdmin is a web-based application written in PHP intended to handle the administration of MySQL over Web. That means it will be no problem, where you are and which OS you have installed on the current machine, you can do the administrative work anywhere. PhpMyAdmin supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc., while you still have the ability to directly execute any SQL statement. Making a backup or restoring a MySQL server is also very simple with this tool.

#### Installation

Installation is very easy on a Debian/Ubuntu machine, but you should have PHP scripts installed on it (see section [20.4](#)).

```
$ sudo aptitude install phpmyadmin
```

#### How to use phpMyAdmin

##### Login

To log into the system, you should enter `http://127.0.0.1/phpmyadmin/` into the location bar of your browser and go. You need to enter the username and password that you have used on MySQL server to log into the platform. For example, username (`root`) and password (`password`)

##### Administration

After a successful login, you can begin with the administrative works. Figure [20.3](#) shows the main page for managing a MySQL Server.

##### tool bar

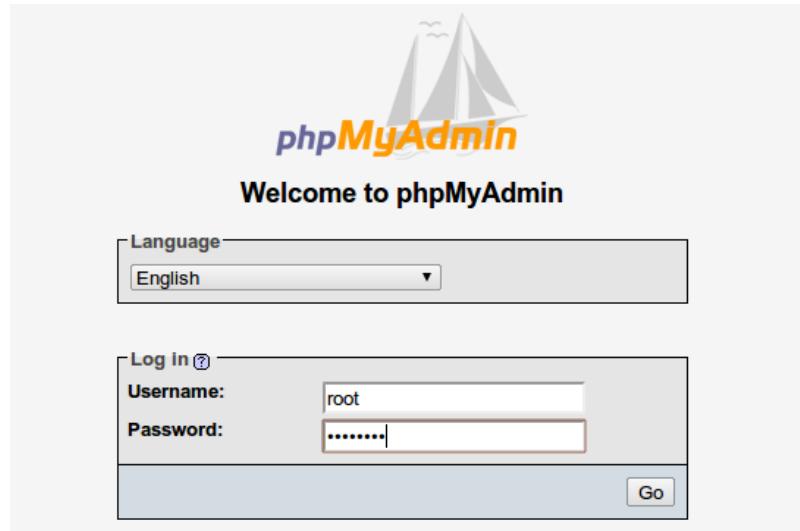


Figure 20.2: Login site of phpMyAdmin

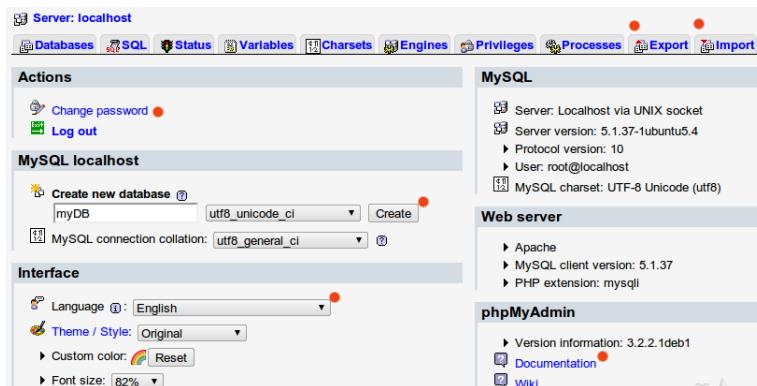


Figure 20.3: Home page of phpMyAdmin

At the top of the main page there is a *tool bar* for the most common administrative jobs on MySQL server. After you choose one database from **Databases** tab, you can execute **SQL** queries on the server and check the current status of MySQL on the tabs **Status**, **Variables**, **Charsets**, **Engines**, **Privileges**, and **Processes**. Of course, you can also make a database backup and restore it with **export** and **import**.

In the middle of the page, you can create a new database "myDB" with **Create**. PhpMyAdmin has localization support, and changing the language of the interface is very easy. You just need to choose the language from **Interface-Language**.

At the right side of this page, you can find general information about MySQL database server, Web Server, and some very useful links like **Documentation**, **Wiki** for phpMyAdmin. To learn more about phpMyAdmin, please follow the link to the **Documentation**.

## 20.4 PHP

PHP: (**P**H<sub>P</sub> **H**ypertext **P**reprocessor) is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP.

PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies.

PHP can be embedded into HTML with the start tag <?php and the end tag ?>. If you use Apache as web server PHP sections are automatically interpreted, if the file has the .php extension.

### 20.4.1 Installation

```
$ sudo aptitude install libapache2-mod-php5 php5-mysql
```

### 20.4.2 A Server-side Scripting Language

Normally, when the browser requests a .html file (static content), the server just sends that file. If the browser requests a .php file (dynamic content), the server reads it, runs any script code inside of it and sends the result the script produces as the response back to the browser. Figure 20.4 shows a specific example to describe how server-side scripts work.

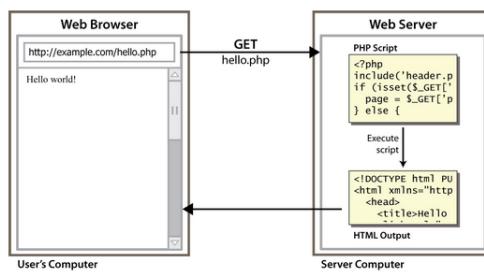


Figure 20.4: A server-side scripting language - PHP

Now let's create a simple but useful PHP script for a Web administrator. By default, we start the scripting in the directory /var/www/

```
$ vi /var/www/info.php
```

Then, put the following content into the script. To test the script, simply type `http://127.0.0.1/info.php` in your browser.

```
<?php phpinfo(); ?>
```

As a Web administrator, you are always going to look for information about PHP scripts or the Web server. The best place to look is in `phpinfo()`. Many of these sections have expanded details that can help you to figure out any problems on the Web server or in the PHP engine.

To learn how to program with PHP, there is a simple on-line tutorial on the official Web site. Here is the link: <http://de2.php.net/manual/en/tutorial.php>

## 20.5 Setting up Web-based open source software with LAMP

### 20.5.1 A internal Knowledge Management System

MediaWiki is a web-based wiki software application used by all projects of the Wikimedia Foundation, and many other wikis. Originally developed to serve the needs of the free content Wikipedia encyclopedia, today it has also been deployed by organizations or companies for internal knowledge management, and as a content management system.

```
$ sudo aptitude install mediawiki
```

#### Installation

**Configuration** The configuration for only one mediawiki on a machine can be easily done by uncommenting the third line of `/etc/mediawiki/apache.conf`

```
# Uncomment this to add an alias.  
# This does not work properly with virtual hosts..  
# Alias /mediawiki /var/lib/mediawiki
```

But as mentioned in the configuration file, this does not work with virtual hosts. The following configuration does not cover this simple option but it works with virtual hosts and also enables updates in minor releases (ex. 1.15.X) happening with normal system updates. Those minor updates only do bug fixes and do not change the database structure. In the following we will link the not changing mediawiki files to a directory, copy the changing files and create a web page configuration file with this directory as `DocumentRoot`. It is possible to repeat this procedure for each additional (virtual host) mediawiki.

```
$ sudo mkdir -p /var/www/balkh-wiki/htdocs
$ sudo cp -a /var/lib/mediawiki/* /var/www/balkh-wiki/htdocs/
```

`ls -l /var/www/balkh-wiki/htdocs/` shows us that most files we have copied are linked to `/usr/share/mediawiki/`. Two files are page specific configuration files (`LocalSettings.php` and `AdminSettings.php`) and link to `/etc/mediawiki/`. We remove those links because we need specific configuration for each wiki. Each wiki needs for example its own database and its own upload folder.

```
$ ls -l /var/www/balkh-wiki/htdocs
$ sudo rm /var/www/balkh-wiki/htdocs/AdminSettings.php /var/www/balkh-wiki/
    htdocs/LocalSettings.php
```

As you can see, two directories `images` and `config` are writable for `www-data` (the username for the Apache web server). `images` is the directory for file uploads and `config` is the directory where the initial web installation process of mediawiki stores the configuration files. For security reasons you can remove the write permission for `config` folder later. The extension folder is only writable by `root`. Thus you will have separate extensions for each wiki installation. If you use a lot of extensions you should think about removing this directory and creating a link to a central extension directory. Not every wiki installation will use every extension: The extensions need to be installed and enabled in each configuration. Extensions are not covered in this documentation<sup>1</sup>.

```
$ sudo cp /etc/apache2/conf.d/mediawiki.conf /etc/apache2/sites-available/balkh-
-wiki
```

Change the new configuration file so that it looks like this:

```
$ sudo cat /etc/apache2/sites-available/balkh-wiki
# balkh-wiki web configuration

NameVirtualHost *:80
<VirtualHost *:80>
    Servername balkh-wiki.local
    ServerAlias balkh-wiki.local *.balkh-wiki.local
    DocumentRoot /var/www/balkh-wiki/htdocs
    <Directory /var/www/balkh-wiki/htdocs/>
        Options +FollowSymLinks
        AllowOverride All
        order allow,deny
        allow from all
    </Directory>
```

---

<sup>1</sup><http://www.mediawiki.org/wiki/Manual:Extensions>

```
# some directories must be protected
<Directory /var/www/balkh-wiki/htdocs/config>
    Options -FollowSymLinks
    AllowOverride None
</Directory>
<Directory /var/www/balkh-wiki/htdocs/upload>
    Options -FollowSymLinks
    AllowOverride None
</Directory>
</VirtualHost>
```

Reload the Apache configuration to activate the virtual host,

```
$ sudo /etc/init.d/apache2 reload
```

add an entry to your `/etc/hosts`

```
127.0.1.2 balkh-wiki.local balkh-wiki
```

and have a look to the page using Firefox (Figure 20.5).



Figure 20.5: Web based mediawiki configuration on virtual host

Fill out the form (Figure 20.6). Replace `<webmaster@yourdomain>` with a real email address of the responsible person. If its is for practicing, use your own email.

As mentioned before, a part of Mediawiki is stored within a database. For security reasons every application gets its own database and database user, if possible. This is the case here, but we haven't created it yet. The Mediawiki web installation will do this for us, if we provide the database root password we set in section 20.3.1 on page 142. (Figure 20.7)

If you read everything and filled out the most important fields as explained and shown above, you can click **Install MediaWiki**. (Figure 20.8)

The screenshot shows the 'Site config' section of the MediaWiki configuration interface. It includes fields for 'Wiki name' (set to 'balikh-wiki'), 'Contact e-mail' (<webmaster@yourdomain>), 'Language' (en - English), 'Copyright/license' (No license metadata selected), 'Admin username' (WikiSysop), 'Password' and 'Password confirm' (both set to a series of dots, with 'Cannot be blank' validation errors), and 'Object caching' (No caching selected). A sidebar on the left lists contributors like Magnus Manske, Brion Vibber, Lee Daniel Crocker, Tim Starling, Erik Möller, Gabriel Wiers, and others.

Figure 20.6: Web based mediawiki configuration

The screenshot shows the 'Database config' section of the MediaWiki configuration interface. It includes fields for 'Database type' (MySQL selected), 'Database host' (localhost), 'Database name' (balikh-wiki), 'DB username' (balikh-wiki), 'DB password' (set to a series of dots, with 'Must not be blank' validation error), and 'DB password confirm' (set to a series of dots). Below these, there are fields for 'Superuser account' (Use superuser account checked), 'Superuser name' (root), and 'Superuser password' (set to a series of dots).

Figure 20.7: Web based MySQL configuration for mediawiki



Figure 20.8: Web based mediawiki configuration - finish installation

If everything was filled correctly, it shows you the success as in figure 20.9. If not, you are redirected to the main configuration page and everything that was wrong or has been left empty is marked red. If so, just add the marked fields and click again on **Install MediaWiki**.

To get access to your wiki you need to copy the `LocalSettings.php` file from the config directory to one level up. Be aware that the web installation assumes that we did the *simple* installation (not copying and removing links) and not the *multiple wikis possible* installation. Thus, it shows



Figure 20.9: Web based mediawiki configuration - installation succeeded

wrong paths. Move, secure and check as follows:

```
$ cd /var/www/balkh-wiki/htdocs/
/var/www/balkh-wiki/htdocs/$ sudo mv config/LocalSettings.php .
/var/www/balkh-wiki/htdocs/$ sudo chown root:root -R LocalSettings.php config/
/var/www/balkh-wiki/htdocs/$ sudo chmod g=r,o= LocalSettings.php
/var/www/balkh-wiki/htdocs/$ ls -l
```

Now only the `images` folder should be writable by `root` and you can “follow this link” (Figure 20.9). Done :-)

## 20.5.2 A Personal Publishing Platform

**WordPress** WordPress<sup>2</sup> is a powerful personal publishing platform, and it comes with a great set of features designed to make your experience as a publisher on the Internet as easy, pleasant and appealing as possible. It offers a freely distributed, standards-compliant, fast, light and free personal publishing platform, with sensible default settings and features, and an extremely customizable core.

Here are the simple steps, how to set up WordPress in a LAMP environment.

First of all, make sure you have LAMP installed.

**Step 1:** download the latest WordPress from <http://wordpress.org/latest.zip>.

**Step 2:** unzip the archive and move the files to DocumentRoot on Apache.

```
$ cp /home/user/Downloads/latest.zip      /var/www/
$ unzip /var/www/latest.zip
$mv latest wordpress
```

**Step 3:** open your browser at <http://localhost/phpmyadmin> and log in using your root and password in order to create a new database `wordpressDB` on MySQL server. Go to the created database and click on privileges and create a new user `demoUser` and give him privileges.

<sup>2</sup><http://www.wordpress.org>

**Step 4:** change the permissions on the directory of WordPress `/var/www/wordpress` and create the configuration file for installation.

```
$ chmod 777 /var/www/wordpress  
$ cp wp-config-sample.php wp-config.php
```

**Step 5:** open your browser at `http://localhost/wordpress`, and you will be prompted to add the database information for a full installation.

```
Database Name: wordpressDB  
User name: demoUser  
Password: *****  
Database Host: localhost
```

**Step 6:** click on **submit** and follow the steps to continue with Installation.

**Step 7:** if the installation is processed successfully, WordPress should be available with default theme at `http://localhost/wordpress` in the browser.

**Note:** if WordPress should be served on a special Web, a virtual host has to be configured properly. The documentation of MediaWiki should be able to give you a reference.

### 20.5.3 A popular Internet Forum Package

`phpBB` is a free flat-forum bulletin board software solution that can be used to stay in touch with a group of people or can power the entire website. With an extensive database of user-created modifications and styles database containing hundreds of style and image packages to customise the board, users can create a very unique forum in minutes.

**Step 1:** download the archive from `www.phpbb.com`

**Step 2:** extract the .zip archive and copy the files into the `/var/www/forum` directory

```
$ unzip /home/<user>/Downloads/phpBB-3.0.7-PL1.zip -d /var/www  
$ mv /var/www/phpBB3 /var/www/forum
```

**Step 3:** begin with installation

```
$ firefox http://localhost/forum
```

Step 4: set the proper permissions on files/directories and continue with installation

```
$ chmod 777 /var/www/forum/cache  
$ chmod 777 /var/www/forum/files  
$ chmod 777 /var/www/forum/store  
$ chmod 777 /var/www/forum/config.php  
$ chmod 777 /var/www/forum/image/avatars/upload
```

Step 5: the system prompts you to create a new database on MySQL.

```
$ mysql -u root -p  
mysql> create database phpbb;
```

Step 6: if the installation is processed completely, rename the install directory and change the permission of config.php. That's all.

```
$ cd /var/www/forum  
$ mv install install.bak  
$ chmod 644 /var/www/forum/config.php
```

**Note:** if phpBB should be served within a special domain or on the Web, a virtual host has to be configured properly. The documentation of installing MediaWiki should be able to give you a reference.



# Apache Web Server

---

## Contents

<b>21.1 Apache Configuration . . . . .</b>	<b>155</b>
21.1.1 Structure of configuration files . . . . .	155
21.1.2 Directives . . . . .	157
21.1.3 Virtual Hosts . . . . .	158
21.1.4 Authentication with .htaccess . . . . .	160
<b>21.2 Apache with SSL/TLS . . . . .</b>	<b>161</b>
<b>21.3 User Web Directory . . . . .</b>	<b>166</b>
<b>21.4 Filter for Compression . . . . .</b>	<b>166</b>
<b>21.5 CGI on Apache . . . . .</b>	<b>167</b>

---

This chapter will take you from basic installation and configuring (see section 20.2) for simple or multiple Web sites to more advanced techniques like HTTPS, logging, mapping, user web directory and so on.

## 21.1 Apache Configuration

### 21.1.1 Structure of configuration files

The Debian/Ubuntu maintainers have a special structure of arranging the configuration files for Apache 2 which is not documented in the standard Apache documentation<sup>1</sup>. This section should help you get acclimated to the Debian/Ubuntu way of configuring Apache 2.

By default Debian/Ubuntu stores its Apache 2 configuration files in the directory `/etc/apache2`. To have an overview of this directory, simply type this command

```
$ ls /etc/apache2 --group-directories-first -l
```

`httpd.conf`

---

<sup>1</sup><http://httpd.apache.org/docs/>

Normally the main Apache configuration file is called `httpd.conf` in other UNIX-like systems. Although you can find the file also in this directory on Debian/Ubuntu, it is because some other software expects it to exist and will add stuff to it. The configuration on Debian/Ubuntu machine starts with the file `apache2.conf` actually.

#### `apache2.conf`

Debian/Ubuntu maintainers have decided that `httpd.conf` was becoming too big. When the user serves many Web sites with virtual hosts on the same server, the user normally adds the directives of Web server and all virtual host blocks to the same file. A modular approach can grow in an ordered fashion with it.

#### `ports.conf`

contains the `Listen` directives telling the Apache server which IP address and port should be listened. It is recommended to add the `NameVirtualHost` directive here. For example, you want to serve a Website on localhost and port 8080 with name-based virtual host, you are able to configure the binding in the same file. Here is the sample code.

```
# /etc/apache2/ports.conf
Listen 8080
NameVirtualHost *:8080
```

#### `conf.d/`

contains the custom configurations for the system user and the special software. Files in this directory are included as part of the "global" server configuration and will apply to all virtual hosts. For example, you can find here the default file like `security`, and you can add all secure configurations to this special file. You can also create a new file yourself here, which you want to be loaded by the Web server and the vhosts. For example: create a new configuration file and append `ServerName 127.0.0.1` to this new file. You should reload Apache to make it available.

```
$ sudo echo ServerName 127.0.0.1 >> /etc/apache2/conf.d/myconfig
```

#### `sites-available` and `sites-enabled`

Debian/Ubuntu maintains Virtual hosts just like modules. Each Web site you want to serve gets its own configuration file with `<VirtualHost>` directive. The `<VirtualHost>` contains all the directives only for that Web site.

As default, Apache is still configured to have one virtual host, the Default Virtual Host named `000-default`. The number ensures that the default virtual host is the first one loaded by Apache. You should always ensure that the default virtual host is the first one loaded.

```
/etc/apache2/sites-available/*  
/etc/apache2/sites-enabled/*  
/usr/sbin/a2ensite  
/usr/sbin/a2dissite
```

#### mods-available and mods-enabled

The Apache web server uses a great modular architecture. You can add or remove functionality as dictated by your requirements. Normally the user can add the configurations to httpd.conf to load the modules. Debian/Ubuntu maintainers create two non-standard directories: /etc/apache2/mods-enabled and /etc/apache2/mods-available to distribute the configurations. Whenever you install an Apache module from repository, the module will store one or two files into the mods-available directory. The `module.load` contains the Apache Load directive to load the module into your web server. The optional `module.conf` file contains additional configuration directives necessary for the operation of the module.

```
/etc/apache2/mods-enabled/*.load  
/etc/apache2/mods-enabled/*.conf  
/etc/apache2/mods-available/*.load  
/etc/apache2/mods-available/*.conf  
/usr/sbin/a2enmod  
/usr/sbin/a2dismod
```

Installing a module from repository makes it available on your server, but does not automatically activate the module on your server. To activate the module, use the `a2enmod` command; to disable the module, use the `a2dismod` command.

#### 21.1.2 Directives

How Apache treats the different requests for the different pages is defined in directives, which structure the configuration files.

NameVirtualHost <sup>2</sup> <i>ip-addr [:port]</i>	Normally the IP address of the server. Can also match to all IP addresses: *. Must exactly match <VirtualHost> directive.
<VirtualHost <i>ip-addr [:port]</i> > <directive <i>a</i> > </VirtualHost>	Contains directives only for the specified virtual host.
ServerName <i>servername.tld</i>	Specifies the servername (must be set also in dns system)
ServerAlias *. <i>servername.tld</i>	Specifies aliases for the servername (pointing at the same site)
DocumentRoot <i>/path/to/htdocs</i>	Where is your webpage stored?
<Directory <i>/path</i> > <directive <i>b</i> > </Directory>	Contains directives only for the specified directory.
AllowOverride <i>keyword</i>	Defines whether .htaccess files are evaluated or not. Keywords can be: All (evaluated and every directive is allowed) / None (not evaluated) / Limit (evaluated and allows use of the directives controlling host access) / ...
...	...

### 21.1.3 Virtual Hosts

If you want to have a single web server system to support more than one Web site, you must configure Apache with virtual hosts. The concept of Virtual Hosts refers to the practice of running multiple Web sites on a **single** machine. Virtual hosts can be "IP-based", meaning that you have a different IP address for every virtual host, or "name-based", meaning that you have multiple different names running on the same IP address. The fact that they are running on the same physical server is not apparent to the end user.

**Name-based virtual host** If there is no specific reason to choose IP-based virtual hosts, you should commonly use Name-based virtual hosts to provide the service on the web server.

If you have only one IP address on the machine, but you want to support more than one Web site on the system. Follow these steps to set up a Name-based virtual host.

1. mapping
2. binding
3. configuring
4. activation
5. reloading and testing

#### Mapping

<sup>2</sup>The convention that words between < > have to be replaced does not apply for this table, because it is used to mark the directives. Instead everything written in *italic* has to be replaced.

To serve Web site with domain name "www.tuberlin.org" on localhost(127.0.0.1), the domain name should be mapped to the correct IP address.

```
# /etc/hosts  
127.0.0.1      www.tuberlin.org      tuberlin
```

### Binding

When Apache starts, it binds some ports and IP addresses which have been defined in `ports.conf`. Type the following directives in order to let Apache listen on port 80 and use Name-based virtual hosts.

```
Listen 80  
NameVirtualHost *:80          # Name-based virtual host on port 80
```

### Configuring

For the Web site `www.tuberlin.org` one demo page should be created in `/var/www/`. Let's create a new directory for the Web site and an index page for a simple test.

```
$ mkdir tuberlin.org && vi tuberlin.org/index.php
```

```
# /var/www/tuberlin.org/index.php  
# It can be very annoying, if you copy code directly from PDF document.  
<?php echo "welcome to demo page for www.tuberlin.org.";?>
```

The next step is to create a `<VirtualHost>` block for each different host that you would like to serve. The argument to the `<VirtualHost>` directive must match a defined `NameVirtualHost` directive.

```
# set up a virtual host in /sites-available/ directory  
$ sudo vi /etc/apache2/sites-available/tuberlin.org
```

the content can be like the following sample code.

Listing 21.1: Sample code of Name-based virtual host

```
<VirtualHost *:80>  
  ServerAdmin webmaster@tuberlin.org
```

```

ServerName www.tuberlin.org
ServerAlias tuberlin

DocumentRoot /var/www/tuberlin.org
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory /var/www/tuberlin.org/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
</Directory>

ErrorLog /var/log/apache2/error.log

# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
LogLevel warn

CustomLog /var/log/apache2/access.log combined
</VirtualHost>

```

**Activation** The next step is to enable the configured virtual host, simply type

```
$ sudo a2ensite tuberlin.org
```

#### Reloading and testing

after the virtual host is enabled with the following command, the testing can be done by entering the URL `www.tuberlin.org` into the location bar of a browser.

```
$ sudo /etc/init.d/apache2 reload
```

**IP-based virtual host** You have different IP addresses assigned to the server and you want to support one Web site on each IP address. This can be achieved by the machine having several physical network cards, or by the use of virtual interfaces which is supported by most modern operating systems.

Section 21.2 shows an example how to set up a secure Web server with IP-based virtual host.

#### 21.1.4 Authentication with .htaccess

To provide basic authentication to web pages requiring login name and password `.htaccess` files can be used and must be stored in the directory you want to protect. (example)

```
AuthType Basic
AuthName "<this_is_my_test_system>"
AuthUserFile /var/www/<webpage>/htdocs/.htusers
Require user <testuser>    # or Require valid-user
```

With the following command you can create an appropriate .htusers file:

```
$ sudo htpasswd -c /var/www/<webpage>/htdocs/.htusers <testuser>
```

## 21.2 Apache with SSL/TLS

### Introduction

As we know, normal web traffic is sent unencrypted via the Internet. That is, anyone with access to the right tools can monitor all of that traffic. Obviously, this can lead to problems, especially where security and privacy is necessary, such as in *credit card data* and *bank transactions*. The Secure Socket Layer is used to encrypt the data stream between the web server and the web client (the browser).

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide security for communications over networks such as the Internet. TLS and SSL encrypt the segments of network connections at the Application Layer to ensure secure end-to-end transit at the Transport Layer. TLS is also the name of a working group of the Internet Engineering Task Force, but in this article TLS refers to the protocol, not the working group.

There are a few key ingredients you will need to use with Apache to secure your Web server: **OpenSSL**, **mod\_ssl**, and **root** access to the server.

**mod\_ssl** This module provides strong cryptography for the *Apache web server* via the *Secure Sockets Layer* and *Transport Layer Security* protocols by the help of the Open Source SSL/TLS toolkit *OpenSSL*, which is based on SSLeay from Eric A. Young and Tim J. Hudson.

**openSSL** OpenSSL is a command line toolkit for using secure sockets layer encryption on a server and can be acquired from [openssl.org](http://openssl.org). This tool works with Apache module mod\_ssl in carrying out SSL-related tasks.

## Step by Step: configure SSL under Apache

In this section, we want to secure the Web site `www.tuberlin.org` on an Apache Web Server with a self-created Certificate step by step:

**Step 1:** You should have Apache 2, mod\_ssl, openSSL installed on the machine. If not, type the following commands:

```
$ sudo aptitude install apache2 libapache-mod-ssl  
$ sudo aptitude install openssl
```

**Step 2:** Create local private/public key for SSL private/public key encryption system with name `server.key`. This key will be used in step 2 to create certificate signing requests. The created RSA private key `server.key` is a 1024 bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text. The command will always prompt you to enter a pass-phrase and store the key directly in `server.key`. If you forget the pass-phrase or it is lost, `server.key` will be useless.

At first, we change the permission as `root`, and create a new directory for storing all certificate-related files.

```
root@desktop:~/ sudo su  
root@desktop:~/ cd /root  
root@desktop:~/ mkdir certificate && cd certificate  
root@desktop:~/certificate# openssl genrsa -des3 -out server.key 1024  
  
Generating RSA private key, 4096 bit long modulus  
.....++  
.....++  
e is 65537 (0x10001)  
Enter pass phrase for server.key:  
Verifying - Enter pass phrase for server.key:  
  
root@desktop:~/certificate# ls -l  
total 4  
-rw-r--r-- 1 root root 3311 2010-06-19 17:05 server.key
```

**Step 3:** Using `server.key` to create a certificate signing request `server.csr`. During the generation of CSR file, you will be prompted for several pieces of information. These are X.509 attributes of standard certificate.

*One of the prompts will be for "Common Name (e.g., YOUR name)". It is important that this field is filled in with the fully qualified domain name of the server to be protected by SSL. If the website to be protected will be https://www.server.com, then enter www.server.com at this prompt.*

```
root@desktop:~/certificate# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.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) [AU]:DE

State or Province Name (full name) [Some-State]:Berlin
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TU Berlin
```

**Step 4:** using private key `server.key` to sign the generated Certificate Request `server.csr` and this certificate should last 365 days.

*A certificate signing request (also CSR or certification request) is a message sent from an applicant to a certificate authority in order to apply for a digital identity certificate. The CSR contains information identifying the applicant (such as a distinguished name in the case of an X.509 certificate). The corresponding private key is not included in the CSR, but is used to digitally sign the entire request. The CSR may be accompanied by other credentials or proofs of identity required by the certificate authority, and the certificate authority may contact the applicant for further information. If the request is successful, the certificate authority will send back an identity certificate that has been digitally signed with the private key of the certificate authority.*

```
root@desktop:~/certificate# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.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) [AU]:DE
State or Province Name (full name) [Some-State]:Berlin
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TU Berlin
    Organizational Unit Name (eg, section) []:ziik Admins Training Center
Common Name (eg, YOUR name) []:tuberlin.org
Email Address []:webmaster@tuberlin.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

root@desktop:~/certificate# cat server.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIE2zCCAsMCAQAwgZUxCzAJBgNVBAYTAkRFMQ8wDQYDVQQIEwZCZXJsaW4xDzAN
â€¢ ...
```

```
gTiFrX63rTn9XJn/NIoHhFABfFr80o99cd5SZphJpj02w+Jnf0i6qIbUG2AudC8=
-----END CERTIFICATE REQUEST-----
```

**Step 5:** Removing the password from the server.key file and keep a copy of server.key with password protection as backup.

```
root@desktop:~/certificate# openssl rsa -in server.key -out
server.key.insecure
Enter pass phrase for server.key:
writing RSA key
root@desktop:~/certificate# mv server.key server.key.secure
root@desktop:~/certificate# mv server.key.insecure server.key
root@desktop:~/certificate# ls -l
total 16
-rw-r--r-- 1 root root 2021 2010-06-19 18:01 server.crt
-rw-r--r-- 1 root root 1760 2010-06-19 17:37 server.csr
-rw-r--r-- 1 root root 3243 2010-06-19 18:10 server.key
-rw-r--r-- 1 root root 3311 2010-06-19 17:05 server.key.secure
```

**Step 6:** Copy certificate related files into the apache configuration directory (It is recommended to create /etc/apache2/ssl).

```
root@desktop:~# mkdir /etc/apache2/ssl
root@desktop:~# cp /root/certificate/server.key /etc/apache2/ssl
root@desktop:~# cp /root/certificate/server.crt /etc/apache2/ssl
root@desktop:~# ls -l /etc/apache2/ssl /
total 8
-rw-r--r-- 1 root root 2021 2010-06-19 19:16 server.crt
-rw-r--r-- 1 root root 3243 2010-06-19 19:16 server.key
```

**Step 7:** Turning on SSL engine and enabling/loading the SSL module on Apache 2.

```
root@desktop:~# a2enmod ssl
Enabling module ssl.
See /usr/share/doc/apache2.2-common/README.Debian.gz on how to configure SSL and
create self-signed certificates.
Run '/etc/init.d/apache2 restart' to activate new configuration!

root@desktop:~# vi /etc/apache2/ports.conf
```

You should add this line to the file ports.conf. You dont need to reload/restart Apache now, because we will make further changes to the configuration files in the next steps.

```
NameVirtualHost *:443
```

**Step 8:** Setting the reference to .crt and .key file in the SSL configuration, which you have stored in /etc/apache2/ssl.

```
root@desktop:~# cp /etc/apache2/sites-available/default-ssl
/etc/apache2/sites-available/tuberlin.org.ssl
root@desktop:~# a2ensite tuberlin.org.ssl
Enabling site tuberlin.org.ssl.
Run '/etc/init.d/apache2 reload' to activate new configuration!
root@desktop:~# vi /etc/apache2/sites-available/tuberlin.org.ssl
```

Now update the content of vhost file `tuberlin.org.ssl` with the following lines and restart Apache server with `/etc/init.d/apache2 restart`:

```
ServerName www.tuberlin.org
DocumentRoot /var/www/tuberlin.org

...
<Directory /var/www/tuberlin.org/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>
...
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
```

**Step 9:** Testing the Web site by browsing with the URL <https://tuberlin.org>.

You should be greeted with a warning dialog if you are using the self-signed certificate. Acknowledge the dialog and the page will continue to load, protected by SSL. The status bar of your browser should be graced by the 'lock' icon, which signifies the page is protected via SSL. This is all there is to it!

**The easy way** OpenSSL comes with a handy script, called `CA.pl`, which simplifies the process of creating keys. To get more HOW TO informations, read the on-line documentation.<sup>3</sup>

<sup>3</sup><http://www.openssl.org/docs/apps/CA.pl.html>

### 21.3 User Web Directory

On multi-user systems, each user on thin-clients can be permitted to have a web site in their home directory using the `UserDir` directive. Visitors to a URL `http://example.com/username/` will get content out of the home directory of the user “`username`”, out of the subdirectory specified by the `UserDir` directive.

For example: a registered user `sampleuser` on Server `user.cs.tu-berlin.de` or `130.149.17.156`, the user’s web content should be available at `http://130.149.17.156/~sampleuser/` or `http://user.cs.tu-berlin.de/~sampleuser/`.

At first, enable the `UserDir` module in Apache 2 by processing this command:

```
$ sudo a2enmod userdir
```

Then create a new folder `public_html` in the home directory and add a line to the file `/etc/apache/httpd.conf`.

```
$ mkdir -p public_html
$ sudo vi /etc/apache2/httpd.conf

# add this line to httpd.conf
UserDir public_html
```

Now the web content in `/home/anyuser/public_html` is available via HTTP. For example: `http://localhost/~anyuser`

### 21.4 Filter for Compression

The `mod_deflate` module provides the DEFLATE output filter that allows output from the Web server to be compressed before being sent to the client over the network i.e. the Internet.

First of all, we need to enable the module `deflate` in Apache 2 with:

```
$ sudo a2enmod deflate
```

The following sample configuration can be used in any vhost configuration file for individual virtual hosts on Apache. If you want to make the compression system-wide available, put the sample configuration in `/etc/apache2/conf.d`.

```
$ sudo touch /etc/apache2/conf.d/compression
$ sudo vi /etc/apache2/conf.d/compression

# this is a sample configuration for mod_deflate
<Location />
    <IfModule mod_deflate.c>
        # place filter 'DEFLATE' on all outgoing content
        SetOutputFilter DEFLATE
        # exclude uncompressible content via file type
        SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png|rar|zip)$ no-gzip

        # properly handle requests coming from behind proxies
        #<IfModule mod_headers.c>
        #    Header append Vary User-Agent env=!dont-vary
        #</IfModule>

        # Properly handle old browsers that do not support compression
        BrowserMatch ^Mozilla/4 gzip-only-text/html
        BrowserMatch ^Mozilla/4\.0[678] no-gzip
        BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
    </IfModule>
</Location>
```

## 21.5 CGI on Apache

Using CGI programs is one of the simplest ways to provide dynamic content Web sites with dynamic content. CGI programs can be written in any programming language. Although CGI is no longer the preferred mechanism for generating dynamic content, it is the simplest and understanding how CGI works is a great help in understanding how the more complex dynamic content providers work. Other dynamic content providers, such as PHP, provide many of the same functions as CGI programs, but typically execute faster than CGI.

**Enabling a CGI directory** A CGI directory is usually enabled by default when you install Apache. But if you want to add more directories for CGI uses, the `ScriptAlias` does this work for you.

Add the following lines to one of the virtual host configuration files on the Apache server. For example, we add the sample code to the configuration file `000-default` for `localhost`.

```
# a sample configuration in /etc/apache2/sites-enabled/000-default

...
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    AddHandler cgi-script .cgi .py .pl
    Order allow,deny
    Allow from all
    AllowOverride None
```

```
</Directory>
...
```

As shown in the configuration, we can add CGI programs to the CGI directory `/usr/lib/cgi-bin`. Now add a new file `helloworld.cgi` with this content:

```
#!/usr/bin/perl -w
use strict;
use CGI;
print "content-type: text/html\n\n";
print "<b>hello world with perl</b>.\n";
```

All the CGI programs should be executable in the system. If the CGI program is written in C, it has to be compiled at first. The solution to this problem is to make sure that our Perl script itself is executable:

```
$ sudo chmod a+x /usr/lib/cgi-bin/helloworld.cgi
```

To test that CGI is set up correctly, enter the URL `http://localhost/cgi-bin/helloworld.cgi` in the browser.

For more information about CGI, read the on-line tutorial <sup>4</sup> and test the scripts in the course.

---

<sup>4</sup><http://www.cgi101.com/book/>

# MyPHPAdmin: Building a web-based administration interface

---

## Contents

<b>22.1 The Administration Panel Modules . . . . .</b>	<b>169</b>
22.1.1 A static HTML site with form . . . . .	169
22.1.2 First php site (declaring variables) . . . . .	170
22.1.3 Second php site (working with variables) . . . . .	171
22.1.4 Third php site (hidden fields) . . . . .	172
22.1.5 Fourth php site (including java script) . . . . .	172
22.1.6 Fifth php site (php functions) . . . . .	173
22.1.7 Sixth php site (validating user input) . . . . .	174
22.1.8 Seventh php site (calling external programs) . . . . .	175
22.1.9 Eighth php site (securing cmd calls) . . . . .	176
22.1.10 Ninth php site (the helper class) . . . . .	176
22.1.11 Tenth php site (the navigation) . . . . .	177
22.1.12 Expect - Introduction . . . . .	177
22.1.13 User management with expect . . . . .	178
<b>22.2 CSS Basics (an excursion) . . . . .</b>	<b>180</b>
22.2.1 CSS Syntax . . . . .	180
22.2.2 IDs and Classes as Selectors . . . . .	181

---

Within this chapter the development and the progress of the MyPHPAdmin System is reflected. The intention of this projects was to give an impression of what is behind of web-based administration interfaces, to gain an insight into php basics, to learn how to automate the daily tasks and finally to combine system and web development in one project.

Where the development of web-based administration interfaces requires a very detailed knowledge of the underling processes, the usage (mostly) is abstracted in a way that also non IT related staff (administration) is able to use that application in a productive way.

## 22.1 The Administration Panel Modules

### 22.1.1 A static HTML site with form

- file: 01-static-html.html

- static html site
- form is included
- form contains field to read username (user input)
- **how to process the user input?**
- **how to create input sensitive response?**
- **how to handle (input) errors?**

### 22.1.2 First php site (declaring variables)

- file: 02-basic-php.php
- (dynamic) php site with one variable
- Variables in php are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.
- Variable assignment:

```
<?php

// Assign the value 'Bob' to $var1
$var1 = 'Bob';

// Reference $var1 via $var2.
$var2 = &$foo;

// Assign $var2 the value of $var1
$var3 = $foo;

$var2 = "My name is $var1";

// $var2 is altered, $var1 is altered too, $var3 is untouched
echo $var1; echo $var2; echo $var3;

?>
```

- variable scope

```
<?php
$a = 1; /* global scope */

function test()
{
    echo $a; /* reference to local scope variable */
}

test();
?>
```

- global variables

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

- global variables II

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

Sum();
echo $b;
?>
```

- “superglobal” variables do not require "global" (e.g. GLOBALS, POST, etc.)
- variable should contain the username (or whatever comes from the user input)
- submitted content (after form is submitted) is written into variable
- the variable content is shown back to the user
- **how to deal with the user input in more detail?**
- **how to catch invalid user input (e.g. empty fields)?**

### 22.1.3 Second php site (working with variables)

- file: 03-basic-php.php
- now, variable should be checked if it contains any content
- check variables if they are empty (returns true or false)

```
empty($variable)
```

- context sensitive output to the user, notice if there is no content
- content is shown back to the user, with exception case
- **but now, it still looks ugly, having this empty answer field at first visit!**
- **how to build the output depending on the users' input?**
- **how to recognize if the user really has submitted the form?**

#### 22.1.4 Third php site (hidden fields)

- file: 04-basic-php.php
- form includes a hidden input field

```
<input type="hidden" name="formsend" value="process" />
```

- use php to check if the hidden field is set

```
if(isset($_POST['formsend'])) { ... }
```

- if the field is set, display the computed response (name)
- else display a message to the user and display the form again
- **now, how can we prevent the user from submitting an empty form?**
- **when this should happen?**

#### 22.1.5 Fourth php site (including java script)

- file: 05-basic-php.php
- first, we can check this on client side
- request comes only to server if it contains a value
- include a small Java script

```
<script type="text/javascript">

function checkForm() {
    if ( document.forms['setname'].elements['name'].value.length == 0 ) {
        alert('Please enter a name');
        return false;
    }
    return true;
}

</script>
```

- scripts runs (observes) the html document's elements
- if the length of the value of the indicated element is zero
- display an alert and quit executing by returning “false”
- attach the script to the form submit action

```
<form action="..." onSubmit="return checkForm()" name="setname">
```

- **phpAdminConsole will contain many modules**
- **for best maintenance we have to prevent redundancy**
- **how to make the code more structured / reusable and less redundant?**

#### 22.1.6 Fifth php site (php functions)

- file: 06-basic-php.php
- encapsulate reoccurring tasks into functions
- setup and introduce php functions
- try to make these function as general / generic as possible (parameters)

```
function foo($arg_1, $arg_2, /* ... , */ $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
```

- recursive functions (CLI php script)

```
#!/Applications/MAMP/bin/php5.3/bin/php
<?php

function recursive($arg) {
    global $value;
    if($arg < 100) {
        $value++;
        recursive($value);
    }
}

$value = 1;
recursive($value);
echo "Result: $value";
?>
```

- flow of control should be located very central (not across the whole code)
- how to catch and handle invalid user input?
- e.g. what happens if ... ?

```
• <script type="text/javascript">
    alert('I am a stupid script!!!!');
</script>
```

### 22.1.7 Sixth php site (validating user input)

- file: 07-basic-php.php
- use trim to: Strip whitespace (or other characters) from the beginning and end of a string

```
string trim ( string $str [, string $charlist ] )
$trimmed = trim($text);
```

- use stripslashes to: un-quote a quoted string

```
string stripslashes ( string $str )
echo stripslashes($str);
$array = stripslashes_deep($array);
```

- use strip\_tags to: remove any tags from the user input

```
string strip_tags ( string $str [, string $allowable_tags ] )
echo strip_tags($text);
```

```
<b>Hello</b>
will become Hello
```

- what if

```
<script type="text/javascript">
    alert('I am a stupid script!!!');
</script>
```

- now, how can we start creating the Administration Panel???

### 22.1.8 Seventh php site (calling external programs)

- file: 08-advanced-php.php

- **What do we need?**

- a website that controls the daily tasks
- website has to interact with the system
- PHP Site has to call an external program (in our case, e.g. a script)
- PHP Site has to evaluate the return value / output of that program
- PHP Site has to give feedback to the user

- **eventually problems / issues:**

- Maybe the PHP site has to call this program with sudo privileges
- the web server user has to have the right privileges
- these privileges have to be as exact as possible (small, on point)
- edit /etc/sudoers to do so ...

```
• www-data ALL=NOPASSWD: /var/www/php/scripts/myLSScript
```

- exec() executes given command(s)

```
• string exec (string $command [, array &$output [, int &$return_var ]])
```

```
• $cmd="$shellscript " . $_POST['directory'];
```

- `exec($cmd,$output,$status);`

- `if ($status == 0) { echo $output; } else { echo "ERROR"; }`

- output only contains **stdout**, how to also get the error messages?
- the script has to redirect stderr into stdout (php site determines the status)
- **what happens if:**

```
/home && cat /etc/passwd
```

- and by the way, “it looks ugly”

### 22.1.9 Eighth php site (securing cmd calls)

- file: 09-advanced-php.php
- use `escapeshellcmd` to: escape any characters in a string that might be used to trick a shell command into executing arbitrary commands

```
string escapeshellcmd ( string $command )
$e = escapeshellcmd($userinput);
$f = escapeshellcmd($filename);
```

- introduce *css* to define the layout
- yeah, looks good now but ...
- we mix code with layout specific commands
- the website’s code is too long, and for the next site I don’t want to do this all again (code and layout) ...

### 22.1.10 Ninth php site (the helper class)

- file: 10-advanced-php.php
- include a helper class for
- all the reoccurring tasks
- all general / common layout issues

```
if ((include 'helper.php')) {  
    echo 'helper v0.1 imported';  
}  
else {  
    echo 'error: helper v0.1 not imported';  
    return;  
}
```

- `class helper { ... }`

- `$helper = new helper();`

- `$helper->writeHeader();`

- extract all frequently used functions to that class and try to make them as generic as possible
- we can use that class to reuse all components - so we can build many sites
- if we will have many sites, we want to integrate basic navigation, too

### 22.1.11 Tenth php site (the navigation)

- file: 11-advanced-php.php
- include a new generic function in the helper class

- `public function writeNavigation($rules) { ... }`

- `$rules = new array(array('one','one.html'),array('two','two.html'));`

- With all that - now let's start to integrate basic user management module (create a new user)

### 22.1.12 Expect - Introduction

- Expect is a program that "talks" to other interactive programs according to a script. Following the script, Expect knows what can be expected from a program and what the correct response should be.
- Example:

```

#!/usr/bin/expect -f

set send_human {.1 .3 1 .05 .3}

send -h "What is 5*5/25 * 9 :\n"

set timeout 10

expect "^[9]\n" {send -h "good, you are right\n"} \
    "\[0-9]" {send -h "ohhh noooooo!\n"} \
    "\[a-zA-Z]" {send -h "sorry, this is no number!\n"} \
    timeout {send -h "too late, sorry\nbye\n"}

```

### 22.1.13 User management with expect

- file: 12-advanced-php.php
- module has to be build like the previous one
- but, the called script now has to interact with the system in a dialog
- we need two scripts, where the first calls the second using the "expect" command

```

#!/bin/bash

USER=$1
PASSWORD=$2
CONFIRM=$3
USERSHELL=$4
ACTIONFILE=/var/www/php/scripts/myUserAddScript
LOGFILE=/tmp/myuserphp.log

exec 4>>$LOGFILE
exec 1>&4
exec 2>&4

echo =====
echo "Date: `date`"
echo "$USER : $PASSWORD : $CONFIRM : $ACTIONFILE"
echo =====

if [ -z $PASSWORD ]; then
    echo "> Info: Please provide a password"
    exit 1
fi

if [ -z $CONFIRM ]; then
    echo "> Info: Please confirm the password"
    exit 1
fi

if [ $PASSWORD != $CONFIRM ]; then
    echo "> Info: Password and Confirmation does not fit"
fi

```

```

        exit 1
fi

if [ -z $USERSHELL ]; then
    echo "> Info: Please provide a shell"
    exit 1
fi

id $USER > /dev/null 2>&1

if [ "$?" -ne "0" ]; then
    echo "> Info: Creating user $USER"
    expect -f $ACTIONFILE $USER $PASSWORD $CONFIRM $USERSHELL
    if [ "$?" -ne "0" ]; then
        RET=$?
        echo "> Something wrong ..."
        exit $RET
    fi
    echo ""
    echo "> Info: Done"
    exit 0
else
    echo "> Info: User $USER is already existing"
    exit 1
fi

```

- the second script, called by the first one by "expect"

```

#!/usr/bin/expect -f

set name      [lindex $argv 0]
set newpass   [lindex $argv 1]
set chkpass   [lindex $argv 2]
set usershell [lindex $argv 3]

spawn useradd -m -s $usershell $name
sleep 2

spawn sudo passwd $name
sleep 2
expect "assword:"
sleep 1
send "$newpass\r"
expect "assword:"
sleep 1
send "$chkpass\r"
expect eof

```

- expect scripts are able to interact with the system
- commands can be spawned (e.g. spawn passwd \$name)

- we can wait (expect) a particular interaction (here password dialog)

```
expect "assword:"
```

- we can submit a value to that dialog

```
send "$newpass\r"
```

## 22.2 CSS Basics (an excursion)

### Styles Solved a Big Problem

HTML was never intended to contain tags for formatting a document. HTML was intended to define the content of a document:

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- Styles were added to HTML 4.0 to solve a problem
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files

All browsers support CSS today.

### 22.2.1 CSS Syntax

A CSS rule has two main parts:

- a selector: The selector is (normally) HTML element that should get a style
- one or more declarations : Each declaration is defined by a property (style attribute) and a value

```
h1 { color: blue; font-size: 14px; }
```

A CSS comment starts with a /\* and end with a \*/ .

**Example:**

```
/*This is a comment*/  
p  
{  
/*This is a comment*/  
color:blue;  
}
```

### 22.2.2 IDs and Classes as Selectors

In addition to setting styles for HTML elements, CSS allows to specify own selectors called "id" and "class".

**IDs:** The id selector is used to specify a style for a single AND unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#" in the css definitions.

```
#contactForm  
{  
text-align:center;  
color:red;  
}
```

**Classes:** The class selector is used to specify styles for a group of (similar) elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for any HTML elements with the same class.

The class selector uses the HTML class attribute, and is defined with a "."

```
.center {text-align:center;}  
.left {text-align:left;}  
p.center {text-align:left;}
```

# Subversion and Trac

---

## Contents

<b>23.1 Subversion . . . . .</b>	<b>183</b>
23.1.1 Introduction . . . . .	183
23.1.2 Installation . . . . .	184
23.1.3 Getting Started with Subversion . . . . .	184
23.1.4 Subversion Commands . . . . .	188
<b>23.2 Trac and Subversion . . . . .</b>	<b>190</b>
23.2.1 Introduction . . . . .	190
23.2.2 Installation . . . . .	190
23.2.3 Configuring the Environments for Trac . . . . .	191

---

This course is for system administrators who need to set up and maintain software development with Subversion software and Subversion repository. The goal of this course is to introduce Subversion and widen the skill of administrators so that they can administer Subversion repository servers and provide the stable code tracking system.

## 23.1 Subversion

### 23.1.1 Introduction

Subversion is a version control system, which allows you to keep old versions of files and directories (usually source code), keep a log of who, when, and why changes occurred, etc., like CVS, RCS or SCCS. Subversion keeps a single copy of the master sources.

Subversion repositories can be accessed (checkout) through many different methods-on local disk, or through various network protocols. A repository location, however, is always a URL. The table describes how different URL schemas map to the available access methods.

Schema	Access Method
file://	direct repository access on the file system
svn://	access via svn protocol to an SVN server
http://	access via WebDAV to Subversion-aware Apache 2 web server
svn+ssh	Same as svn://, but through an SSH tunnel
https://	Same as http://, but with SSL encryption

For more information about the Subversion project:

- Subversion project site <http://subversion.tigris.org>
- Subversion Ebook <http://svnbook.red-bean.com>
- Subversion API Docs <http://svn.collab.net/svn-doxygen/>
- Subversion Repository <http://svn.apache.org/repos/asf/subversion/trunk/>

### 23.1.2 Installation

```
# install subversion
$ sudo aptitude install subversion

# install related package for Web accessing on Apache
$ sudo aptitude install libapache2-svn
```

### 23.1.3 Getting Started with Subversion

#### Local file system

In the following steps, we will demonstrate how to create a simple repository containing some files, and how to check out and commit files.

**Create a local repository** Create a new directory to set up the subversion repository

```
$ mkdir -p ~/svn
```

Create the repository using the **svnadmin** command

```
$ svnadmin create /home/username/svn/repository
```

Check the current status from repository

```
$ svn info file:///home/username/svn/repository
```

Create virtual directories in repository

```
$ svn mkdir file:///home/username/svn/repository/foo -m "foo"
```

Check the log file from repository to view the changes in the past

```
$ svn log file:///home/username/svn/repository
```

**Importing an existing project in repository** Now we have created a new repository. In the following step, we will import some project files from file system in the svn repository.

Import an existing project from file system

```
$ svn import /path/ file:///home/user/svn/repository -m 'Initial import'
```

/path/ should be a path to directory, NOT path to a file!

Import existing files into a virtual directory in repository

```
$ svn import /path/ file:///home/user/svn/repository/itck -m 'Initial import'
```

SVN will create a virtual directory /itck in repository and then import the files in created virtual directory. This will help you to serve different projects in different virtual directories.

**Check out a project from repository** As a member of a development team, we should get a working copy on the local file system from the repository by using `svn co` or `svn checkout`.

```
$ mkdir -p ~/workspace && cd ~/workspace
$ svn co file:///home/user/svn/repository/itck
```

Within a working copy, type the following commands to check the current meta information of your working copy.

```
$ svn ls
$ svn log
$ svn info
$ svn update
```

### Remote accessing with SVN protocol

This section describes setting up a SVN server on an Debian/Ubuntu system and configuring svn for use by a group of developers. For a secure sharing between the team members, it is necessary to allow the members to access the Subversion repositories from a remote location using either the svn or svn+ssh protocol.

To start svn server as in daemon mode (**-d**) as a foreground process (**-foreground**), and to look for repositories in the repository dir that was created earlier (**-r /usr/local/svn/repos**).

```
$ sudo svnserve -d --foreground -r /home/toleuser/svn/repository
```

Now you can access svn server with svn protocol without any authentication.

```
$ svn info svn://127.0.0.1/your_project_name_here
```

To set up a basic authentication with **/repository/conf/svnserver.conf**, uncomment the configuration like this.

*Note: whitespace sensitive, be sure not to leave any whitespace at the start of a line*

```
[general]
anon-access = none
password-db = passwd
```

then you need to add authenticated user and password in **/repository/conf/passwd**.

*Note: whitespace sensitive, be sure not to leave any whitespace at the start of a line)*

```
[users]
iraqadmin = password
toleuser = password
```

### Subversion, WebDav, Apache

To access the SVN repository via WebDAV protocol, you must configure your Apache2 web server.

```
$ sudo aptitude install libapache2-svn
```

Assume that we want to access the SVN via WebDAV ([www.tuberlin.org/svn](http://www.tuberlin.org/svn)): we have configured `tuberlin.org` in the past, and we should only update the configuration by adding this snippet in your `/etc/apache2/mods-enabled/tuberlin.org` file.

```
<Location /svn>
  DAV svn
  SVNPath /path/to/svn_project
  #SVNParentPath /path/to/svn      # alternative
</Location>
```

If we want to limit the access on the repository and allow only the access from the specified users, we can create a password file and configure WebDAV to make a basic authentication with this file.

Create a password file and set the proper permissions.

```
$ sudo touch /etc/subversion/passwd
$ sudo htpasswd /etc/subversion/passwd user_name
```

Update the configuration in `/etc/apache2/mods-enabled/tuberlin.org` with additional content like:

```
<Location /svn>
  DAV svn
  SVNPath /path/to/svn_repository
  AuthType Basic
  AuthName "Subversion Repository via webDAV"
  AuthUserFile /etc/subversion/passwd
  Require valid-user
</Location>
```

To make the update available, you have to reload the Apache configuration. Finally browser Subversion repository in a browser at <http://www.tuberlin.org/svn>.

```
$ sudo /etc/init.d/apache2 reload
$ firefox http://www.tuberlin.org/svn
```

If the the repository has to be secured with SSL, a sample configuration might be like this. *Note: this sample code is not based on the above example that we mentioned.*

```

Listing 23.1: A Sample VHOST Configuration of Subversion Repository via HTTPS

<VirtualHost *:443>
    ServerName svn.tuberlin.org
    <Location />
        DAV svn
        AuthType Basic
        AuthName "svn.tuberlin.org"

        AuthUserFile /opt/svn/svn.tuberlin.org/conf/apache2.passwd
        SVNPath /opt/svn/svn.tuberlin.org/
        Require valid-user
    </Location>
    CustomLog /var/log/apache2/svn.tuberlin.org/access.log combined
    ErrorLog /var/log/apache2/svn.tuberlin.org/error.log

    SSLEngine on
    SSLCertificateFile      /etc/apache2/ssl/server.crt
    SSLCertificateKeyFile  /etc/apache2/ssl/server.key
</VirtualHost>

<VirtualHost *:80>
    ServerName svn.tuberlin.org
    Redirect / https://svn.tuberlin.org
</VirtualHost>

```

### 23.1.4 Subversion Commands

**Getting help** If you are a beginner, you need always to list all subversion commands or to get help on a given command. For example, getting help on subversion command `copy`.

```

$ svn --help
$ svn help copy

```

**Checking out and committing** To make a working copy, we check out the source code from repository with `checkout` or `co`; to commit the changes, we use `commit` or `ci`.

```

$ svn co <repository/any/path>
$ svn ci -m "commit message"

```

**Logging** The information of logging is stored in the directory `.svn` and it is readable with specific tool `svn log`.

```

$ svn log                      # check the logging of a working copy
$ svn log <file name>          # check the logging of a file

```

```
$ svn log -r 4:2          # check the logging of a file with options
$ svn log -l 5            # check the logging of last 5 commit
$ svn log -v              # print the extra meta information
```

**Displaying the differences** Show file diffs between SVN repository and your file changes using GNU file diff format.

```
$ svn diff                # compare BASE and your working copy
$ svn diff <file name>    # compare BASE file and the file in working copy
$ svn diff -r 5:2          # compare reversion 5 and reversion 2
```

**Rolling back the changes** You can roll back all changes and throw away them in the local working copy with `svn revert`, or you can undo the changes in a reversion and roll back the changes in another old reversion with (`svn merge`).

```
$ svn revert               # undo all the changes on the working copy
$ svn revert <file name>  # undo the changes on a file
$ svn merge -r 5:2 <target> # roll back the changes in reversion 2
```

**Resolving the conflicts** For every conflicted file, Subversion places three extra files in your working copy:

**filename.mine** This is your file as it existed in your working copy before you updated your working copy. that is, without conflict markers. This file has your latest changes in it and nothing else.

**filename.rOLDREV** This is the file that was the BASE revision before you updated your working copy. That is, the file that you checked out before you made your latest edits.

**filename.rNEWREV** This is the file that your Subversion client just received from the server when you updated your working copy. This file corresponds to the HEAD revision of the repository.

Here *OLDREV* is the revision number of the file in your .svn directory and *NEWREV* is the revision number of the repository HEAD

If you get a conflict, you need to do one of three things:

1. Merge the conflicted text “by hand” (by examining and editing the conflict markers within the file).
2. Copy one of the temporary files on top of your working file.
3. Run `svn revert <filename>` to throw away all of your local changes.

Once you've resolved the conflict, you need to let Subversion know by running `svn resolved`. This removes the three temporary files and Subversion no longer considers the file to be in a state of conflict. For example: We throw out all the changes on file `info.txt` to resolve the conflict:

```
$ svn revert info.txt
$ svn resolved info.txt
```

**Note:** More practical examples might be found in the course stuffs like slides or handout, such as detailed explanation of resolving the conflict.

## 23.2 Trac and Subversion

### 23.2.1 Introduction

Trac is an open source, web-based project management and bug-tracking tool.<sup>1</sup> The program is inspired by CVSTrac, and was originally named svntrac due to its ability to interface with Subversion. It provides an interface to Subversion (or other version control systems), an integrated Wiki and convenient reporting facilities.

### 23.2.2 Installation

Trac is written in the Python programming language and needs a database like MySQL, SQLite or PostgreSQL. Trac is shipped with a server included, and it is very easy to server it with running the standalone Server. Alternatively we can also configure Trac to run on a Apache Web Server. You can also find many resources about Installation on Linux here.<sup>2</sup>

The following tutorial gives you a simple way to install and set up trac on a Debian/Ubuntu machine(On-line reference for Ubuntu users<sup>3</sup>). First of all, be sure you have the following packages installed on the machine.

```
$ python -V          # you should have python installed, version >2.4
$ mysql -V
$ apache2 -V
$ svn --version
```

Installing all necessary dependencies

---

<sup>1</sup><http://trac.edgewall.org/>

<sup>2</sup><http://trac.edgewall.org/wiki/TracInstall>

<sup>3</sup><http://trac.edgewall.org/wiki/TracOnUbuntu>

```
$ sudo aptitude install python-setuptools python-subversion  
libapache2-mod-python libapache2-svn
```

- `python-setuptools` : *Python Distutils Enhancements*
- `python-subversion` : *Python bindings for Subversion*
- `libapache2-mod-python` : *Python-embedding module for Apache 2*
- `libapache2-svn` : *Subversion server modules for Apache 2*

Installing Trac with `easy_install`

```
$ sudo easy_install trac  
$ trac-admin -v          # current version: trac-admin 0.12
```

Now we have the software installed on the Ubuntu machine, and we will begin with the configuration of Trac environment in next chapter.

### 23.2.3 Configuring the Environments for Trac

Assume that we start with a new project. We need Trac system for project management and graphical front-end web interface to SVN repository. The Trac system should be served at `trac.tuberlin.org`. The directories for Trac environment and SVN repository will be created in directory `/opt`.

**Specify a place where Trac can live** Trac uses a directory structure and a database for storing project data. The directory is referred to as the `environment`. The following command is used to create a Trac environment in the specified directory:

```
$ sudo mkdir /opt/trac/trac.tuberlin.org -p  
$ sudo trac-admin /opt/trac/trac.tuberlin.org initenv
```

`"/opt/trac/trac.tuberlin.org"` is the target directory where Trac environment is created.

**Create SVN repository** One of most important functionalities is that Trac provides a Graphical front end to SVN where diffs in files can be obtained. As a example, we create a repository for the demo project in directory `/opt/svn/tuberlin.org`

```
$ sudo mkdir /opt/svn/svn.tuberlin.org -p
$ sudo svnadmin create /opt/svn/svn.tuberlin.org
```

**Create password file for Trac and SVN** In this demo project, we use the same password file for both system: Trac and SVN. We create this password file with:

```
$ cd /opt/svn/svn.tuberlin.org/conf/
$ sudo htpasswd -b -c apache2.passwd toleuser Password
# toleuser is any name that we want to use in the system
# Password is any password used for this user 'toleuser'
```

**Set the proper permissions on files/directories** We have to set the pertinent permissions on directories where Trac and SVN live, so that Apache server can read and write on them.

```
$ cd /opt
$ sudo chown -R root:www-data trac svn
$ sudo chmod -R 2770 trac/trac.tuberlin.org svn/svn.tuberlin.org
# 2770: auto set group ID
```

**Configure the virtual host for Trac** As known, we want to serve the Trac system with domain name `trac.tuberlin.org`, a virtual host must be configured on Apache. The following is a sample code that can be used for this:

Listing 23.2: A Sample Configuration

```
<VirtualHost *:443>
    ServerName trac.tuberlin.org
    DocumentRoot /opt/trac/trac.tuberlin.org/
    <Location />
        SetHandler mod_python
        PythonHandler trac.web.modpython_frontend
        PythonInterpreter main_interpreter
        PythonOption TracEnv /opt/trac/trac.tuberlin.org/
        PythonOption TracUriRoot /

        # basic authorization, use the same password file like svn
        AuthType Basic
        AuthName "trac.tuberlin.org"
        AuthUserFile /opt/svn/svn.tuberlin.org/conf/apache2.passwd
        Require valid-user
    </Location>

    # separated log file
    CustomLog /var/log/apache2/trac.tuberlin.org/access.log combined
    ErrorLog /var/log/apache2/trac.tuberlin.org/error.log
```

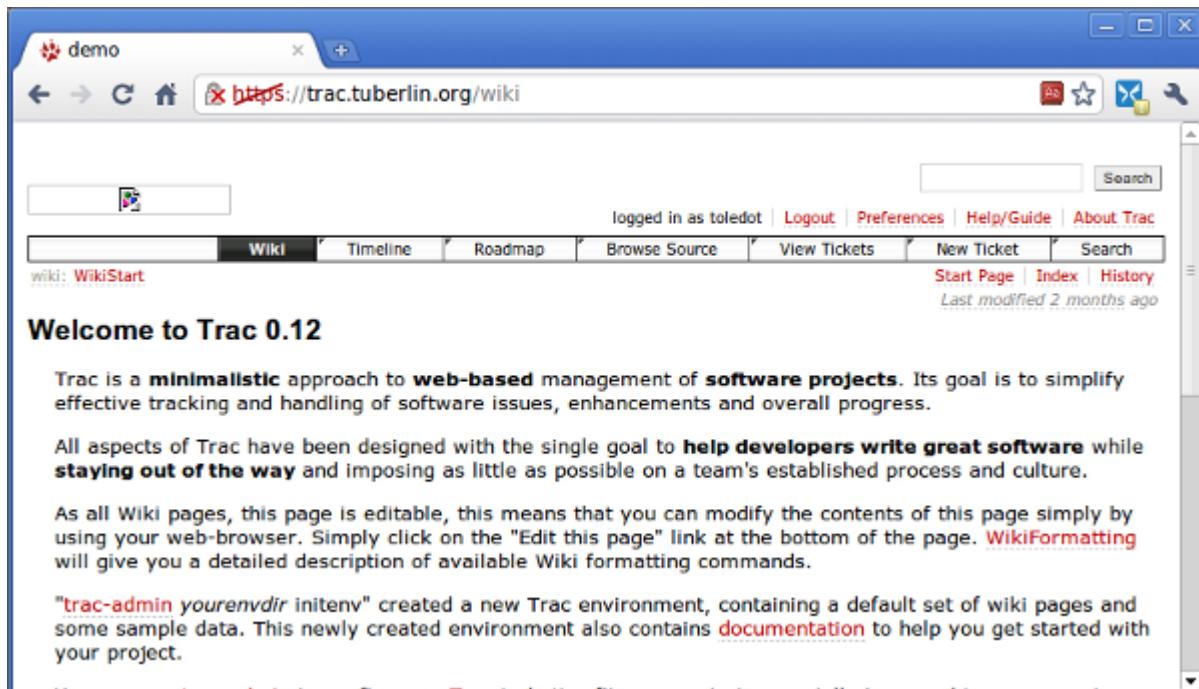


Figure 23.1: A Demo Trac system via HTTPS

```

SSLEngine on
SSLCertificateFile      /etc/apache2/ssl/server.crt
SSLCertificateKeyFile  /etc/apache2/ssl/server.key
</VirtualHost>

<VirtualHost *:80>
    ServerName trac.tuberlin.org
    Redirect / https://trac.tuberlin.org
</VirtualHost>

```

**Test Trac in browser** After update of Apache configurations, we have to reload or restart the service and test the Trac system with any browser on the machine(Figure 23.1).

```

$ sudo /etc/init.d/apache2 restart
$ firefox https://trac.tuberlin.org

```

**Administering via. web interface** Perhaps it will be a little complex to administer the trac with `trac-admin`. A official tutorial about `trac-admin` is here.<sup>4</sup> But the work of administration can be easier, if we active the web-admin plugin. At first we should specify which user will have `trac-admin` permission. This user should be created at first in the password file for TRAC and SVN and then specify the user as `TRAC_ADMIN` in trac console.

<sup>4</sup><http://trac.edgewall.org/wiki/TracAdmin>

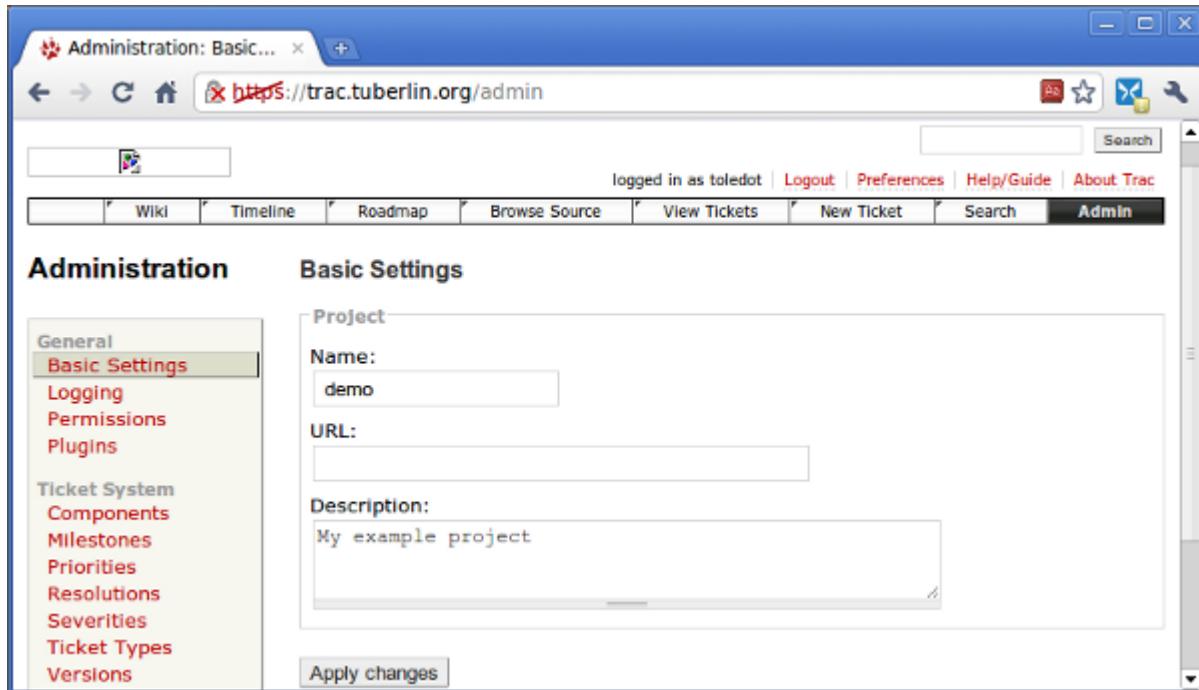


Figure 23.2: Administration of Trac via Web Admin Plugin

```
$ sudo trac-admin /opt/trac/trac.tuberlin.org/
```

If the path of Trac is assigned properly, we are able to enter the Trac console. Here we add the user `toleuser` to the group `TRAC_ADMIN` with permission `add` and list the permissions of that user with command `permission list`:

```
trac [/path/trac]> permission add toleuser TRAC_ADMIN
trac [/path/trac]> permission list toleuser
```

After we have signed the proper permissions to user `toleuser`, the integrated plugin `Web Admin` is activated. Refresh the page and we will see a new tab `Admin` in the right upper side of the page. Click on the `Admin` tab and you are able to modify the most important features of Trac system directly. A view of administration via `Web Admin Plugin` is shown in Figure 23.2.

**Change default logo with yours** As you have detected, the logo on top of the page is not displayed. To change the logo(Figure 23.3), store your logo image in `<TracEnvironment>/htdocs/yourLogo.png`. At last, update the Trac configuration file `<TracEnvironment>/conf/trac.ini` with this line:

```
src = site/yourLogo.png
```

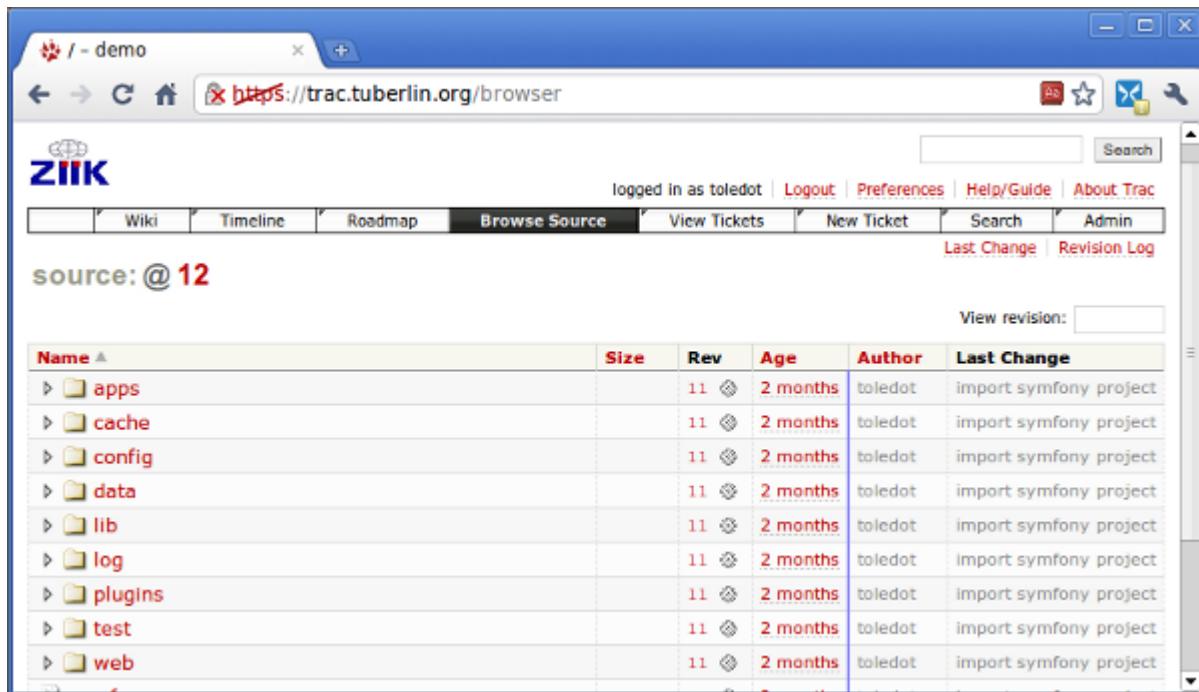


Figure 23.3: Trac With New Logo Image

Note: TracEnvironment in our case is `/opt/trac/trac.tuberlin.org`.

**Access SVN repository via Trac** Before we haven't added a repository to Trac, the tab **Browse Source** is not displayed on the top menu bar in Trac. Now we can update Trac configuration to add a default repository.

```
sudo vi /opt/trac/trac.tuberlin.org/conf/trac.ini
```

Update the pertinent lines with the following content:

```
repository_dir = /opt/svn/svn.tuberlin.org
repository_sync_per_request = (default)
repository_type = svn
```

Reload the page and a new tab **Browse Source** should be displayed on the top menu bar (Figure 23.3). If the source code is not synchronized in Trac, have a try with this command:

```
$ sudo trac-admin /opt/trac/trac.tuberlin.org resync
```

Congratulations, the Trac system is set up on the Apache Web Server completely. The accompanying tutorial in Trac system should be detailed enough to be used as HOW-TO guidance, we will not mention any more.

```
$ firefox https://domain_your_trac_system/wiki/TracGuide  
# i.e. https://trac.tuberlin.org/wiki/TracGuide
```

# Backup Strategies

---

## Contents

<b>24.1 Backup strategies . . . . .</b>	<b>197</b>
24.1.1 Why a backup is necessary . . . . .	197
24.1.2 Developing a backup strategy . . . . .	198
24.1.3 using different backups for different data . . . . .	198
24.1.4 Storage Media . . . . .	199
24.1.5 Cycles . . . . .	200
<b>24.2 Backup methods and tools . . . . .</b>	<b>201</b>
24.2.1 File-based backups . . . . .	203
24.2.2 Backing up whole partitions or disks (Images) . . . . .	203
24.2.3 Backup whole file systems . . . . .	207
24.2.4 Backing up databases . . . . .	207
24.2.5 Backup with archiving tools . . . . .	208
24.2.6 Backup with rsync . . . . .	209
24.2.7 Backup with complex large scale software . . . . .	209
<b>24.3 Example scripts . . . . .</b>	<b>210</b>

---

We begin this chapter with some general backup philosophy, followed by a discussion of the most commonly used backup devices and media(their advantages, weaknesses and costs.) Next, we discuss Linux backup and archiving commands and tools. Finally, the reference to complex and powerful backup software is given for self-study. There are also some small example scripts for backup that have been covered during the lessons.

## 24.1 Backup strategies

### 24.1.1 Why a backup is necessary

The answer to this question seems pretty obvious: at the most, the information stored on the computers is worth more than the computers themselves. Protecting this information is one of the system administrator's most important tasks.

But the effort to restore a broken system corresponds reversely to the efforts one is making to backup a system. Since there are several possible reasons for a system to crash, one has to ponder which of those reasons are likely.

Possible reasons for losing data could be:

- the hardware gets broken (e.g. hard disks since there are moving parts)
- hardware theft (e.g. a laptop, USB flash drives)
- an attacker corrupted the system (either with physical access or remotely e.g. computer viruses)
- users/administrators make mistakes (removal of important files, wrong configured services)
- fire, floods, earth quakes

One consequence derives directly: it is desirably to separate the system and the backup physically and if possible even spatially.

#### 24.1.2 Developing a backup strategy

The backup strategy depends on the very scenario the considered system is part of. Different strategies and the use of different tools arise from different situations, requirements, conditions and possibilities. Therefor an administrator has to find sufficing answers to the key questions, which are:

- What should be backed up? System files, personal documents, databases, configurations files, or the whole file systems?
- Where to backup? Which backup media are available to be used? Which backup media are needed to be considered?
- How much space is needed? (depends on what exactly to be backed up)
- How often and in which cycles should the backup be made?
- How? Which tools could be used?

#### 24.1.3 using different backups for different data

The question what to backup, leads often to different approaches. We are just mentioning different parts of the system that have different backup requirements.

##### system directories

On the one hand the system consists of the core libraries, files and programs that are mostly installed from a Linux/UNIX web repository or other installation media. On the other hand there is often a lot of customization, configuring or installation of additional software done. While one can relatively easy reinstall software from the repositories, it is crucial to configure the services and customize the installation, done mainly in `/etc`, `/boot`, `/usr/local` or `/opt`. Therefor if space

is short, one can abstain from backing up the whole system directories but backup only the customizations. If there is no lack of space, it is more convenient to backup the whole system, since it is easier to recover the system.

For Windows systems it is often necessary to backup disks as a whole or at least the system partition (there are special tools for such an endeavor, like **Acronis**)

### Users' data

The users' data is unique and contains often the projects which users are working on and the configured preferences for the programs. This data must not get lost or corrupted and should be backed up at least daily. Users can make mistakes and for example remove important configuration files. In case a user becomes aware of having removed important files after a couple of days, it provides often more convenience to be able to restore data backed up before a specified time. Therefor one wants to have cycling backup (see below).

### Shared data, mail- and web-content ...

Mostly the same conditions and requirements hold for these, but the data needs even more space and may be partly organized in databases.

### Databases

A database is a collection of information that is organized such that it can easily be accessed, managed, and updated. Databases are used widely and in many applications. It is very important to keep them uncorrupted and consistent, hence it is inevitable to backup them to restore them in case of a corruption.

#### 24.1.4 Storage Media

We just want to mention the most common media here:

**Magnetic tape** Magnetic tape has long been the most commonly used medium for bulk data storage, backup, archiving, and interchange. Tape has a good capacity/price ratio. There are many formats, many of which are proprietary. Tape is a sequential access medium, so even though access times may be poor, the rate of continuously writing or reading data can actually be very fast. Some new tape drives are even faster than modern hard disks. A principal advantage of tape is that it has been used for this purpose for decades (much longer than any alternative) and its characteristics are well understood.

**Hard disks** The capacity/price ratio of hard disk has been rapidly improving for many years. This is making it more competitive with magnetic tape as a bulk storage medium. The main

advantages of hard disk storage are low access times, availability, capacity and ease of use. External disks can be connected via local interfaces like SCSI, USB, FireWire, or eSATA, or via longer distance technologies like Ethernet, iSCSI, or Fibre Channel. The main disadvantages of hard disk backups are that they are easily damaged, especially while being transported (e.g., when used as external drive), and that their stability over periods of years is a relative unknown.

**Optical disc** A recordable CD can be used as a backup device. One advantage of CD backups is that they can in theory be restored on any machine with a CD-ROM drive. (In practice, writable CD-ROMs are not always universally readable.) In addition, recordable CDs are relatively cheap. Another common format is recordable DVD. The newer HD-DVDs and Blu-ray Discs dramatically increase the amount of data possible on a single optical storage disk. But the physical lifetime of the optical disk has become a concern as it is possible for some optical disks to degrade and lose data within a couple of years.

**Solid state storage** Also known as flash memory, thumb drives, USB flash drives, Compact Flash, SmartMedia, Memory Stick, Secure Digital cards, etc., these devices are relatively costly for their low capacity, but offer excellent portability and ease-of-use.

**Storage Area Network (SAN)** A SAN is an architecture to attach remote computer storage devices (such as disk arrays, tape libraries) to servers in such a way that the devices appear as locally attached to the operating system. Although the cost and complexity of SANs are dropping, they are uncommon outside larger networks.

The data transfer in a SAN is mainly the transport of blockbased data (like the communication between computer and hard disks is blockbased). Most SANs are using the SCSI-protocol that is mapped over TCP/IP (iSCSI) or Fibre Channel (FCP).

**Network Attached Storage (NAS)** NAS systems contain one or more hard disks, often arranged as logical, redundant storage containers or RAID arrays (redundant arrays of inexpensive/independent disks). NAS removes the responsibility of file serving from other servers on the network.

NAS uses file-based protocols such as NFS (popular on UNIX systems), SMB/CIFS (Server Message Block/Common Internet File System), or AFP (used with Apple Macintosh computers).

**Direct Attached Storage** Direct-attached storage (DAS) refers to a digital storage system directly attached to a server or workstation, without a storage network in between. The main protocols used for DAS connections are ATA, SATA, SCSI, SAS and Fibre Channel.

#### 24.1.5 Cycles

A backup on servers/workstations, that are usually running without being shut down unless there is an error, is often running once a day, usually at a time when the workload is small (e.g. during night or during lunch break). On special systems backups might be run even more often. Backups often are designed such that one is able to restore data from backups older than the most

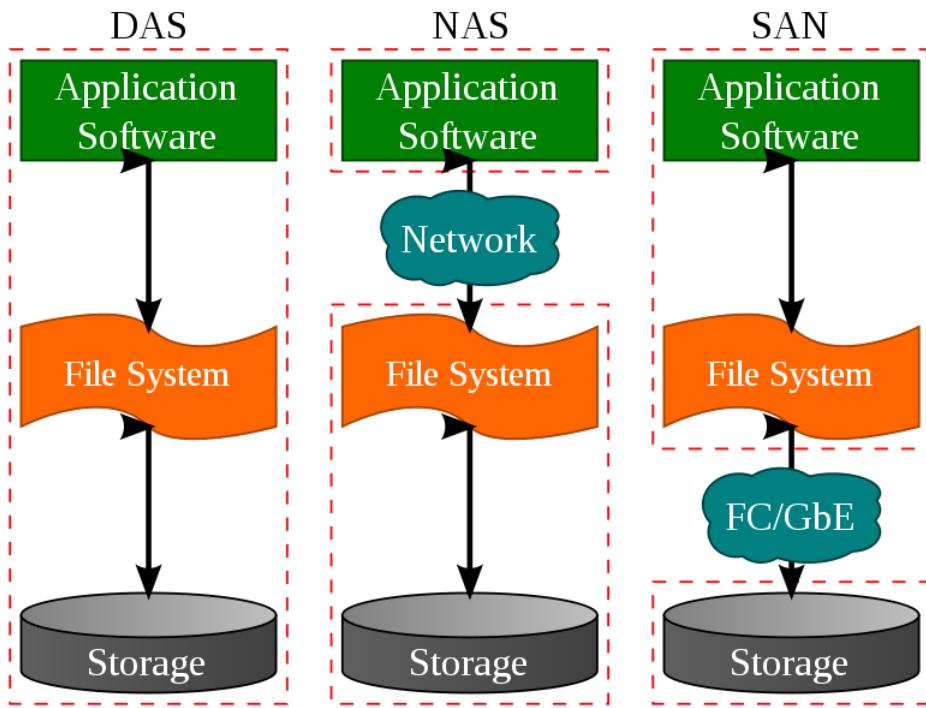


Figure 24.1: Schematic comparison of DAS/NAS/SAN, source: wikipedia

recent. This might be necessary in case some thing went wrong (important directory accidentally removed, but realized only two days later). One possibility to implement that would be running a backup of all data every day (**full backup**), which would take a lot of space on your backup medium. The solution is to backup only the changes to the last backup (**incremental backup**). Another possibility is to backup daily the changes to the last full backup (**differential backup**). This three backup levels are sometimes combined in one cycle.

### Example cycles

- Weekly Cycle: On Sunday **full**, on the other days just **incremental**.
- Monthly Cycle: On the first Sunday of the month **full**, on the other Sundays **differential**, on the weekdays **incremental**.

## 24.2 Backup methods and tools

There is a huge number of possibilities to implement a backup. You can either write your own (shell)-script that uses command line tools or you can use one of the many applications whose use cases reach from occasional backups of a single computer to network based large scale backups of whole server farms. We will only cover the small scale command line/script usage here but mention popular large scale backup software as well.

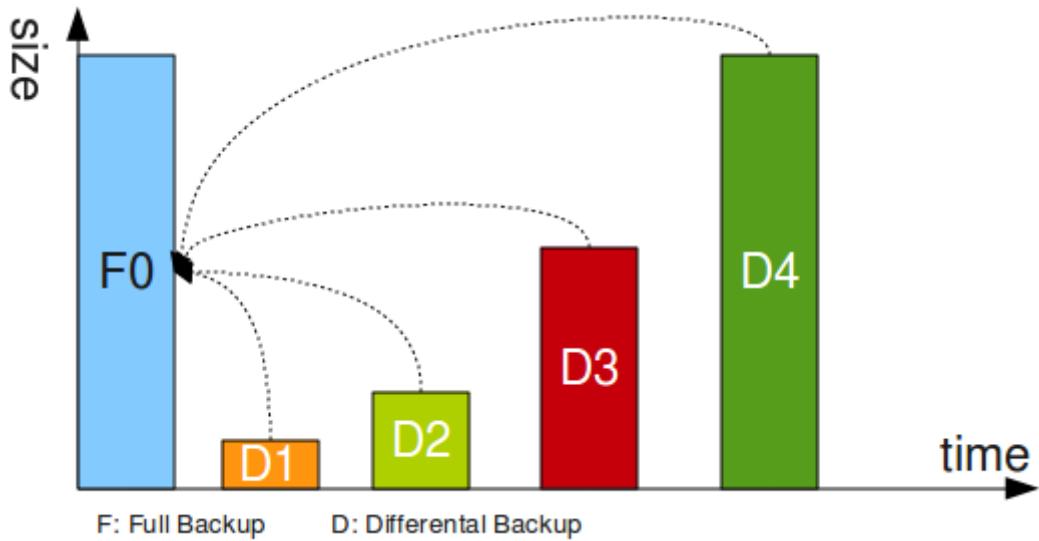


Figure 24.2: Differential Backup

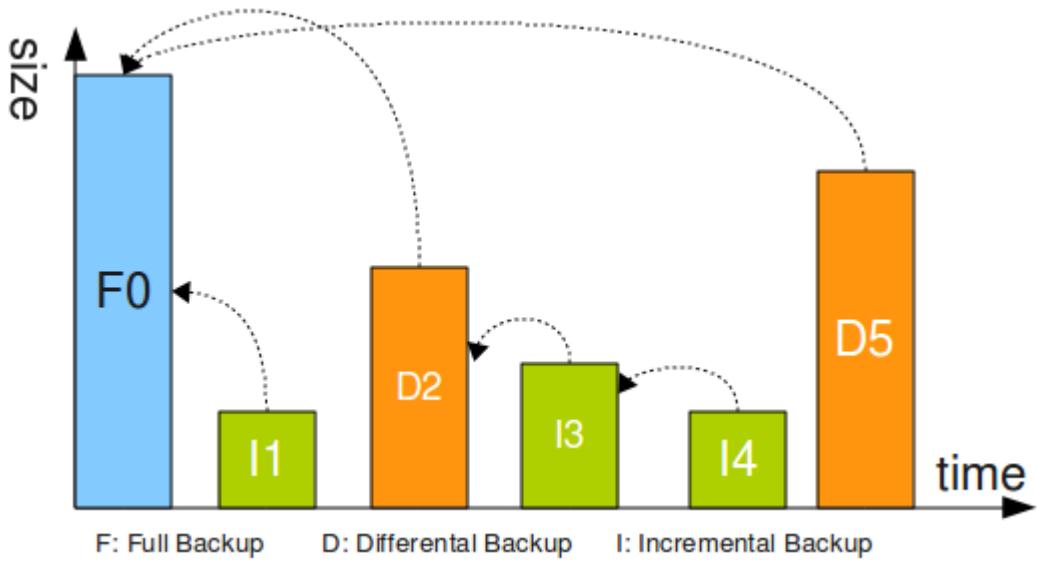


Figure 24.3: Incremental Backup

### 24.2.1 File-based backups

File based backups offer the greatest flexibility, as you can always decide, which files and folders you want to backup or restore. Also, you can easily realize incremental or differential backups.

### 24.2.2 Backing up whole partitions or disks (Images)

Backing up whole partition is relatively easy and you do not have to deal with permissions, attributes, folders etc. because you will operate below the file system layer, directly on the hard disk blocks. This means, you cannot select or exclude particular files or folders, but will always copy the complete contents of a partition. When restoring, you will always reproduce the exact state of the partition, as it was, i.e. you do not even have to format your new partition before restoring the backup. Partition images can be created with several tools:

#### dd

On Linux/UNIX systems you can copy files, partitions and even whole disks with the command `dd`. You maybe have to use this in combination with a Linux Live System (in case you want to copy system partitions), because you should not copy a *mounted* partition with `dd`. Since Linux/UNIX handle disk and partitions as files, the syntax is always the same no matter on which level you are operating:

```
$ dd if=<infile> of=<outfile> [options]
```

An basic example to just copy the contents of a partition (`sda1`) into an ISO file (`diskimage.iso`):

```
$ dd if=/dev/sda1 of=/tmp/diskimage.iso
```

The following example is a bit more advanced, as it implements error skipping on the source media, sets a manual block size for better speed performance and pipes the output through `gzip` to create a compressed partition image file (`sda.img.gz`):

```
$ dd if=/dev/sda conv=sync,noerror bs=64K | gzip -c > /mnt/tmp/sda.img.gz
```

Remember, that the image file must not be written to the same partition that you are backing up as this will likely result in a broken image file. Also remember, that only unmounted partitions should be processed with `dd`.

To restore an image like this, you would use the following command:

```
$ gunzip -c /mnt/tmp/sda.img.gz | dd of=/dev/sda conv=sync,noerror bs=64K
```

`dd` works with most file systems (also Windows file systems, from within a Linux Live CD).

## Clonezilla

Clonezilla is an open source partition imaging application which supports all major file systems (ext2, ext3, ext4, reiserfs, xfs, jfs of GNU/Linux, FAT, NTFS and HFS+ of Mac OS). Therefore you can clone GNU/Linux, MS Windows and Intel-based Mac OS, no matter if they are 32-bit (x86) or 64-bit (x86-64). For these file systems, only used blocks of partitions are saved and restored. For unsupported file systems, Clonezilla is using `dd` for sector-to-sector copies.

Clonezilla is best used from the live CD, which is downloadable from the project website<sup>1</sup>. The following example explains the steps necessary to create a partition image (used Clonezilla version is 1.2.2-31):

After booting the CD (or the ISO image) and selecting the language and keymap, Clonezilla can be started with the selection of the type of operation (Figure 24.4).

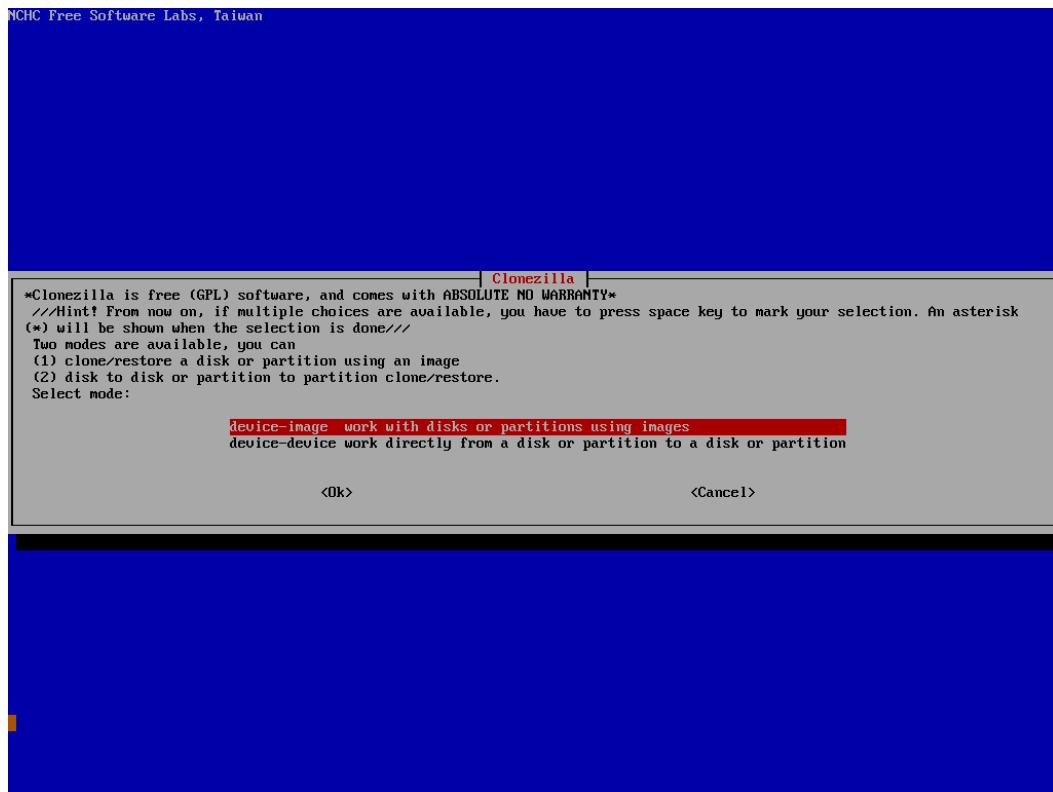


Figure 24.4: Select type of operation

<sup>1</sup><http://www.clonezilla.org>

Two types of operation are possible: “Device to image”, which creates partition or disk image files, or “device to device” which clones a partition or disk directly to another partition or disk on the same host. Typically, select the first option (Figure 24.5).

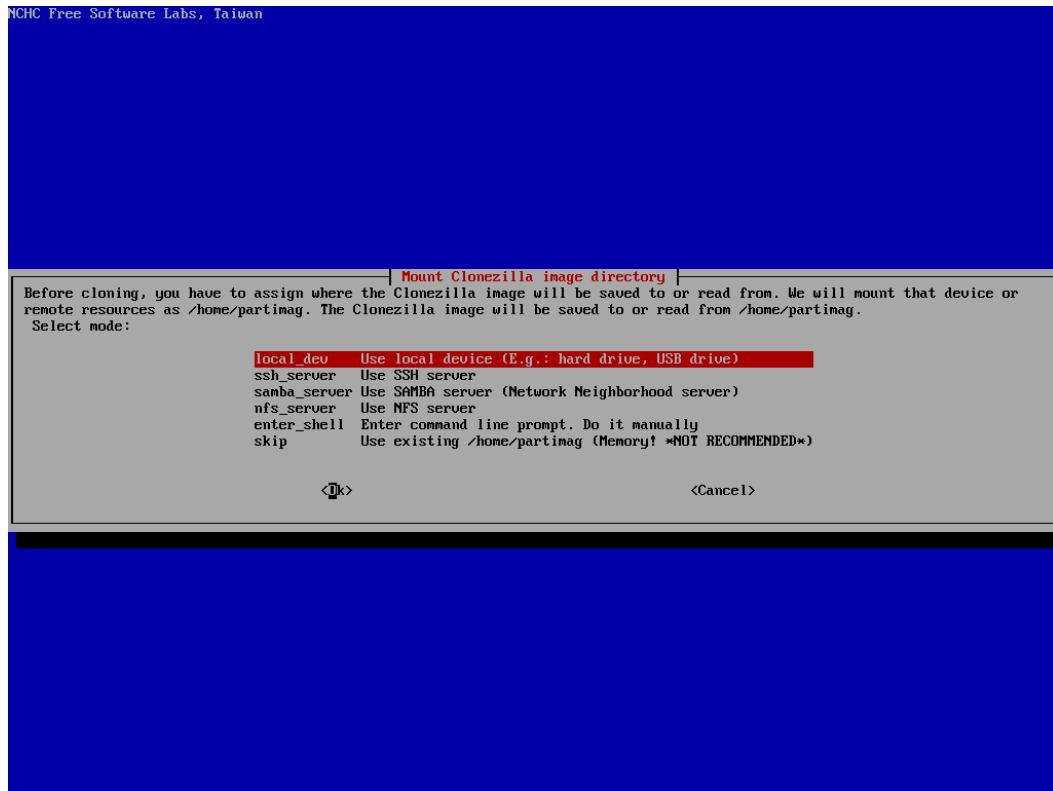


Figure 24.5: Select destination media type

Clonezilla can save the image files to either a local device or to another host over the network via SSH, Samba or NFS. In this example, we pick the first option, “local device” (Figure 24.6).

Therefore, a local partition, which is *not* the partition to be backup up, has to be mounted by Clonezilla for writing the image files to. Select one that fits and select a folder (e.g. “Top\_directory\_in\_the\_local\_device”).

When asked for “Beginner” or “Expert” mode, pick the first one as this sets the most common options (Figure 24.7).

Choose “*saveparts*” for creating an image file of a *partition*. “*savedisk*” creates an image of the *whole hard disk*, including all partitions, which is not what we want to do here now.

In the next dialogue, specify a filename for the image (Figure 24.8).

Then, press up/down and space to select the desired partition. After the summary, press just enter and “y” and again enter to confirm, and the image creation process will begin.

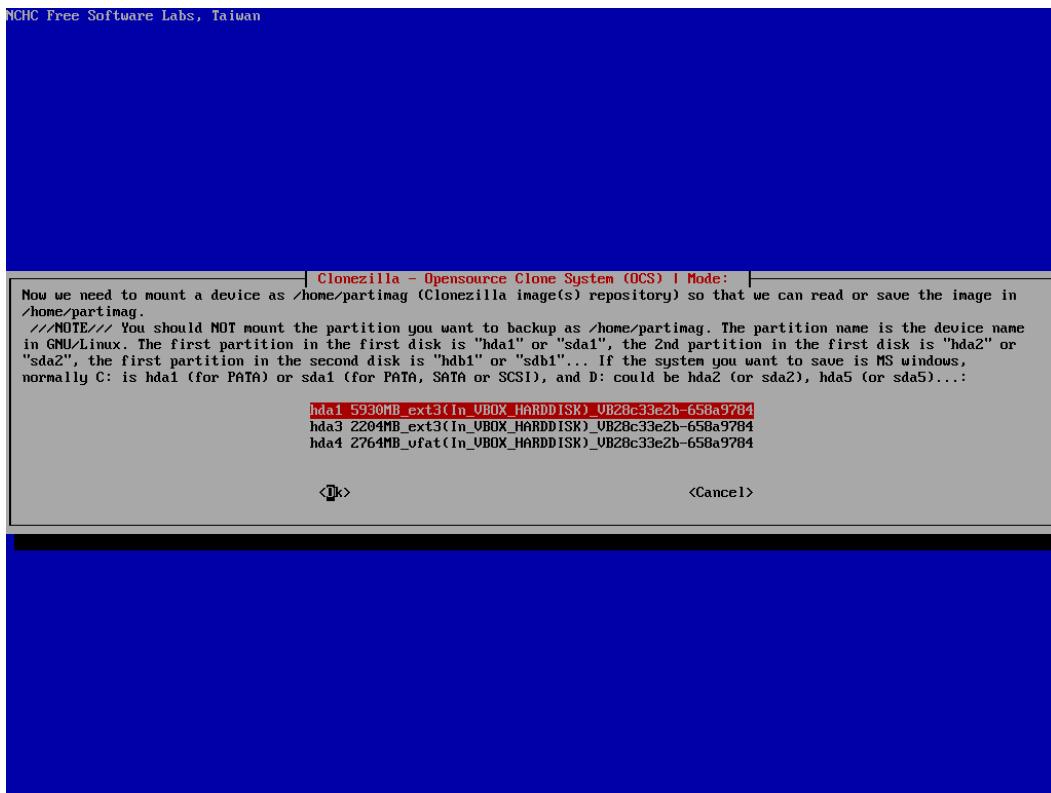


Figure 24.6: Select destination partition

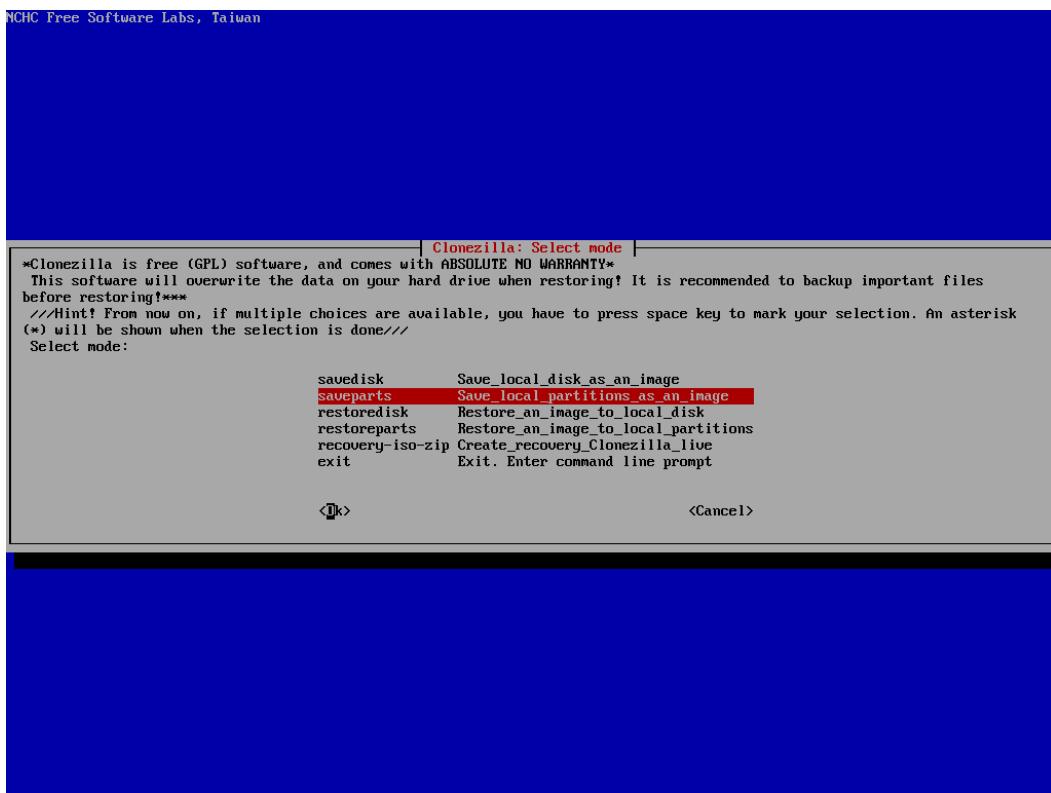


Figure 24.7: Select mode of operation

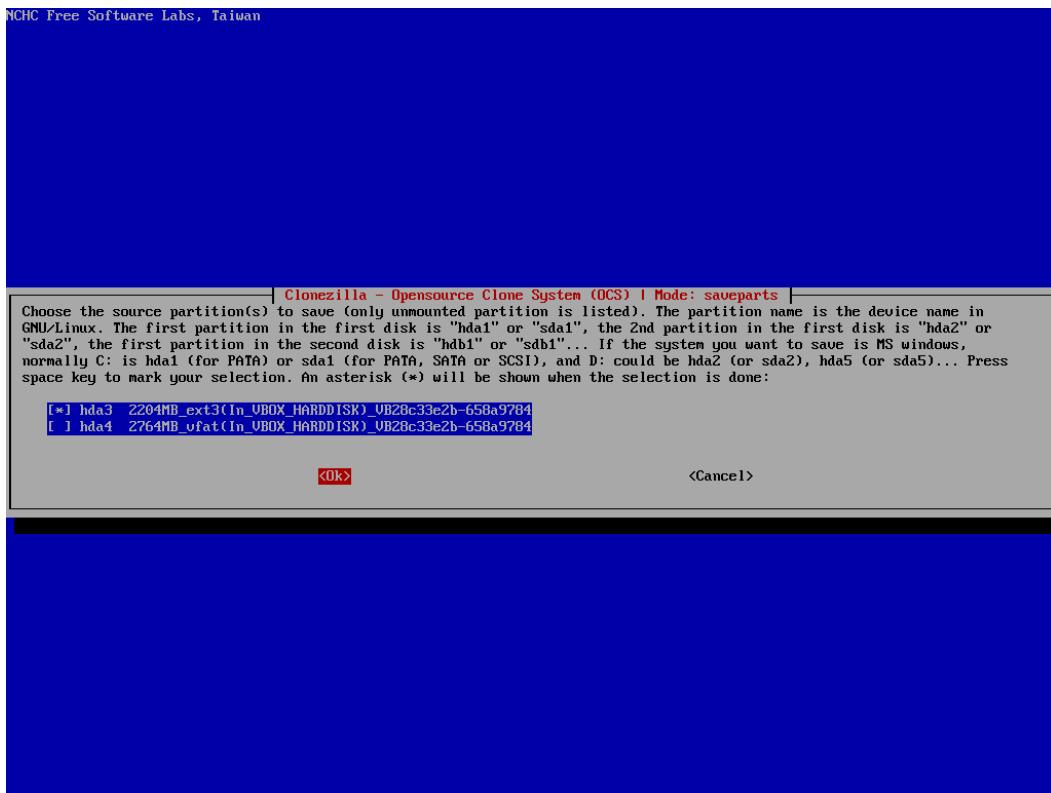


Figure 24.8: Select source partition

## Acronis

Another, quite popular solution for partition imaging is “Acronis True Image”, which is proprietary and only available for Windows (even though it also comes with a Linux-based live CD). [tru 2009]

### 24.2.3 Backup whole file systems

`dump` and `restore` operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories. `dump` is only capable of backing up an entire file system, but cannot backup directories that spans more than one file system.

Both methods can be used over the network as commands `rdump` and `rrestore`

### 24.2.4 Backing up databases

Since most backup tools handle databases as a whole but only the files forming the database one has to dump databases manually. The two major open source database systems MySQL and PostgreSQL bring their own sql-dump-tools:`mysqldump`, `mysqlhotcopy` for MySQL, `pg_dumpall`, `pg_dump` for PostgreSQL.

### 24.2.5 Backup with archiving tools

#### **tar, star**

With **tar** one can archive files or directories. It was developed to write on tapes but can write to files as well (option **-f**). **tar** is available on many systems and in slightly different versions. The standard usage is:

```
$ tar [options] [destination] [files and directories to archive]
```

A backup of a whole directory like */home* works like this:

```
$ tar -[P]cf /backup/backup.tar /home
```

**-c** creates an archive. **tar** removes all the beginning */* by default, so if you want to directly restore from the archive the */* are kept with **-P**.

**tar** can as well update archives with **-u**, which does not work for compressed archives.

Often archives are compressed with **gunzip** or **bzip2** or directly by using **tar** with compression switches the following way:

```
$ tar -czf /backup/backup.tar.gz /home
```

where **-z** stands for the usage of **gunzip** or with **-j** to use **bzip2** instead. One can **tar** even use for incremental backups. With the switch **-N DATE | --newer DATE**. An example usage for an incremental backup archive could be:

```
$ tar -N $(date -d "now 1 day ago" +%Y-%m-%d) -Pcf /backup/backup-1.tar /home/
```

Other useful options for **tar** are: **--exclude=PATTERN** (exclude file that match a pattern) and **-t** to show the content of an archive.

**tar** is not capable of storing access control lists (ACLs) on files. Use **star** for such purpose.

One powerful backup program that uses **tar** is **flexbackup**<sup>2</sup>, that is capable of running full, incremental or differential backups.

#### **cpio**

**cpio** (copy in/out) works similar to **tar**, but is even capable of reading damaged archives. Refer to the man page of **cpio** to learn more.

---

<sup>2</sup>see <http://flexbackup.sourceforge.net/>

### 24.2.6 Backup with rsync

**rsync** is a network protocol as well as a program released under the GPL. The program **rsync** synchronizes file trees (from server to client). The files on the client (destination) are divided into blocks, for each block two checksums (one to easily find the equal blocks, the second to avoid collisions) are built that are transferred to the server (where the source tree is). With help of the checksums the server generates commands how the source file can be generated by blocks of the destination file. Blocks already present on the destination system are only moved. The missing blocks are transferred. Only files that are not yet present on the client are transferred directly. This is an effective way to keep two file system trees synchronous.

**rsync** is able to work over file system borders as well as over network (typically via **ssh**). It works very fast, hence a fast restore is possible. The program is available for many operating systems. It does support compressed data transfer but cannot store compressed files. It is able to preserve ACLs on files.

Basic usage:

```
$ rsync <options> <source> <destination>
```

The most useful options regarding the implementation of a backup are:

- exclude files matching certain patterns: **--exclude=PATTERN** or **--exclude-from=FILE**
- archive (preserve file attributes, device files and symbolic links): **-a**
- delete files that are on the destination file tree but not on the source: **--delete**
- preserve ACLs: **-A**
- **--link-dest=DIR**, hardlink to files in DIR when unchanged, this option is very useful for incremental backups <sup>3</sup>
- **-e**: specify which remote shell to use (e.g. **ssh**)

There are a couple of backup programs based on **rsync**: **grsync** (simple GUI), **flyback** or **rsnapshot**.

### 24.2.7 Backup with complex large scale software

In large networks much more complex and powerful suites are used in order to save, restore and check data in a heterogeneous network. These tools use database back-ends to store information.

<sup>3</sup>see example scripts in section 24.3

**Bacula**<sup>4</sup>

**bacula** has a client/server-architecture and supports a couple of professional backup devices like tape-libraries. It is configurable via a console, a GUI or a web-interface. It uses MySQL, PostgreSQL or SQLite databases as back-end. The different tasks are done by different daemons that are communicating over TCP/IP. To circumvent the problems and limitations of system tools like tar or zip it has its own tools. It can only backup files but not databases.

**Amanda**<sup>5</sup>

**amanda** uses system tools like tar and is capable of backing up databases. Apart from that it is very similar to **bacula**.

### 24.3 Example scripts

The following scripts can be run daily for example with cron.

Listing 24.1: Backup using tar

```
#!/bin/bash

# check for the weekday
WEEKDAY='date +%A'
echo $WEEKDAY

# assign the variables
BACKUP_PATH=/backup

WHAT_TO_BACKUP="foo\bar"

BACKUP_FILE=backup_with_tar_$WEEKDAY

# check whether backup is mounted
# $variable means that we use it (dereference it)
RET=$(cat /proc/mounts | grep $BACKUP_PATH)
echo $RET

if [ -n "$RET" ]; then
    echo 'already mounted'
else
    echo 'not mounted, mounting it now'
    mount $BACKUP_PATH
fi

# start the backup
echo 'starting backup'
tar -cpzf $BACKUP_PATH/$BACKUP_FILE.tar.gz $WHAT_TO_BACKUP

# display the content
```

---

<sup>4</sup>see: <http://www.bacula.org/en/>

<sup>5</sup>see: <http://www.amanda.org/>

```
echo 'show content'
tar -tf $BACKUP_PATH/$BACKUP_FILE.tar.gz

# we have to be careful, therefor we unmount our backup medium
umount $BACKUP_PATH
```

Listing 24.2: Incremental backup using rsync

```
#!/bin/bash

# how many daily backups do we want to store?
NUM_DAILY=7

# Where to backup?
BACKUP_PATH="/backup/"

# directories to backup
# think about using trailing slashes or not
SOURCES="foo/ bar/"

# excluded from backup
EXCLUDES=backup_rsync_exclude.txt

#rsync options
OPT="-av --numeric-ids --stats --exclude-from=$EXCLUDES --link-dest=
$BACKUP_PATH/daily.1"

# check whether backup medium is mounted
RET=$(cat /proc/mounts | grep $BACKUP_PATH)"

if [ -n "$RET" ]; then
    echo 'already mounted'
else
    echo 'not mounted, mounting it now'
    mount $BACKUP_PATH
fi

# rotate out the daily backups
i=$[NUM_DAILY -1]

rm -rf $BACKUP_PATH/daily.$i

while [ $i -gt 0 ]
do
    if [ -d $BACKUP_PATH/daily.$((i-1)) ];then
        echo 'rotating backups'
        echo "mv $BACKUP_PATH/daily.$((i-1)) $BACKUP_PATH/daily.$i"
        mv $BACKUP_PATH/daily.$((i-1)) $BACKUP_PATH/daily.$i

    else
        echo 'creating new backup-dir'
        echo "mkdir $BACKUP_PATH/daily.$i"
        mkdir $BACKUP_PATH/daily.$i
    fi
```

```
i=$[$i - 1]
done

mkdir $BACKUP_PATH/daily.0

# do the actual backup
rsync $OPT $SOURCES $BACKUP_PATH/daily.0

umount the backup medium
umount $BACKUP_PATH
```

# Linux system recovery

---

## Contents

<b>25.1 root password recovery . . . . .</b>	<b>213</b>
<b>25.2 Boot loader re-writing . . . . .</b>	<b>213</b>
<b>25.3 Simple system backup with tar . . . . .</b>	<b>215</b>

---

When a Linux system is not booting properly anymore or shows other symptoms of severe malfunction, there are many possibilities of fixing an installation without reinstalling from scratch. Unlike Windows, full system control is possible under Linux even without booting the installed system and with the right know-how, many serious problems can be fixed without data loss.

## 25.1 root password recovery

If the root password of a Linux system is lost for some reason, it can quite easily be re-set. For that, it is necessary to boot into a different Linux system on the same computer, typically any Linux live system (e.g. from CD or USB device). Once the live system has booted and a terminal with root access is opened, the following procedure can be applied:

1. The partition with the Linux installation to be changed has to be located. The `fdisk -l` command is helpful in this matter, it prints all partitions on all hard disks on the machine, as well as their sizes and partition types.
2. The assumed system partition has to be mounted; e.g.: `mount /dev/sda1 /mnt`
3. With the `chroot` command, the installation on the partition can be “entered”. E.g.: `chroot /mnt`
4. Within the chroot environment, just type `passwd` to set a new root password.
5. `exit` the chroot, `umount` the partition and `reboot`.

## 25.2 Boot loader re-writing

The boot loader is a small “program” on the first 512 bytes of the hard disk (the so called master boot record MBR), which is loaded by the BIOS, when the PC is booting. Under Linux, “grub”

is the most common boot loader. Grub itself offers a number of options for starting (booting) an operating system, and it also has a minimal pre-boot shell-like environment to fix some problems that might occur at boot time. For Linux, grub loads the kernel from the hard disk and starts it.

If the boot loader got corrupted (e.g. after a Windows installation on the same machine) or has to be modified (e.g. due to an added partition with a new operating system installation), this can be done with the `grub` command:

```
$ grub
```

To use grub to write an MBR to the disk, it has to be instructed, on which partition the `/boot` folder resides, because the grub configuration files (as well as the kernel, initrd etc.) are usually in `/boot/grub`. It has to be kept in mind, that grub itself uses a different notation of hard disks and partitions. In general, hard disk names start with “hd” and use only numbers, starting with “0”.

As an example, to write a new boot loader to the first hard disk, and the `/boot` folder is on the first partition of this hard disk, the following commands are necessary:

```
$ grub
grub> root (hd0,0)
grub> setup (hd0)
```

Or, to write the boot loader with the `/boot` folder on the 3rd partition:

```
$ grub
grub> root (hd0,2)
grub> setup (hd0)
```

If you are booting from a live CD in order to restore the boot loader, you have to run grub from within a `chroot` environment (see previous section [25.1 on the preceding page](#)). In order to make grub find the partition information, the `/dev` file system has to be mounted inside the chroot. Also, the `/proc` file system has to be accessible for some applications. Assuming the partition is mounted in `/mnt`, this can be done with the following commands, *before* chroot’ing:

```
$ mount -o bind /dev /mnt/dev
$ mount -t proc proc /mnt/proc
```

## 25.3 Simple system backup with tar

As the Unix philosophy is “Everything is a file”, a complete system backup can be easily done by just copying or achieving all files, i.e. the whole file system tree. As an example, we want to create a tar.gz archive which contains the whole file system tree. Even though, some folders should be omitted, because they are carrying file systems, which are dynamically created by the kernel and not on the hard disk (namely `/proc` and `sys`). Some other folders can also be excluded, just because they are empty or just used as mount points. An example `tar` command to create such an archive would be like this:

```
$ tar -czvp --exclude /mnt --exclude /proc --exclude /sys --exclude /tmp -f /tmp/backup.tar.gz /
```

The “-c” option tells tar to create a new archive, the “-p” option to preserve all file permissions. The “-z” option instructs tar to compress the archive with gzip. The “-v” option just lets tar print all files it is processing on the standard output (terminal).

To exclude the `/tmp` folder is important, because the archive will be written in this folder. Not excluding it could result in a loop, when it tries to include the archive in itself.

To restore from this archive, the following command could be used:

```
$ cd /
$ tar -xvzpf /tmp/backup.tar.gz
```

After this, the excluded folders have to be created, before the system is able to boot again:

```
$ mkdir -p /mnt /proc /sys /tmp
```



# Email Server

---

## Contents

---

<b>26.1</b>	<b>Introduction</b>	217
<b>26.2</b>	<b>Components</b>	218
26.2.1	MTA	218
26.2.2	LDA, IMAP/POP3 Server	219
26.2.3	MUA	219
<b>26.3</b>	<b>Installation</b>	219
26.3.1	Postfix	219
26.3.2	Dovecot	221
26.3.3	MySQL	223
26.3.4	Dovecot SASL	228
26.3.5	PostfixAdmin	229

---

This chapter gives an introduction into how an email server is working and of which components it typically consists.

An example configuration is then discussed which includes basic email sending and receiving, virtual users and a webmail interface.

## 26.1 Introduction

Whether you work for a small organisation or just use email to keep in touch, tailoring an open source email server to your needs is a great way to regain control over your mail. Using any of the big, free providers makes having an email account easy, but leaves you little that can be centrally monitored, configured or enforced. If you work for a small organisation you may also have concerns about privacy or want more control over your account options. You could buy a domain name and get dedicated email hosting from a specialised provider, but even that doesn't give you as much flexibility as it could. Thanks to free software, however, even a small group can get more bandwidth or disk space and make use of full custom filtering, privacy protection and many other features âŞ all with the smallest possible costs and maintenance effort.

## 26.2 Components

The following is an overview diagram of the components of a typical email server system and how they are interconnected:

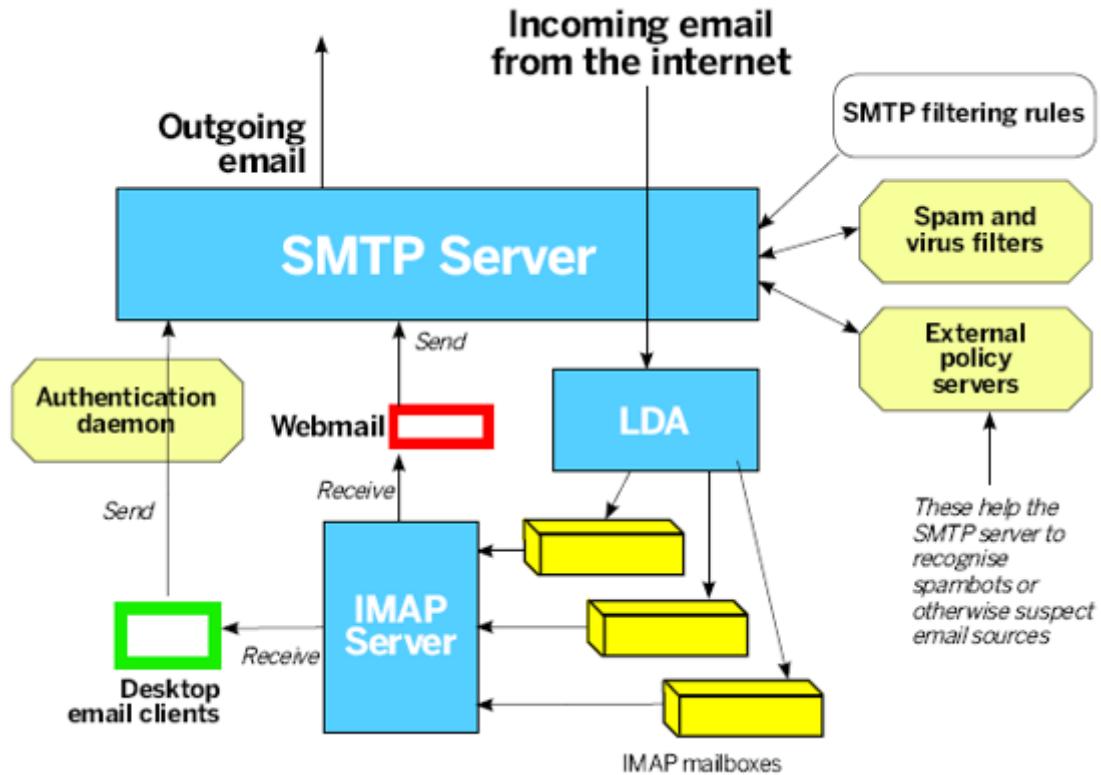


Figure 26.1: Diagram of mail server components (Source: <http://www.tuxradar.com>)

As a minimum, these components are required for a basic email server system:

### 26.2.1 MTA

The **MTA** is the Mail Transfer Agent and is responsible for relaying the email users are sending from their desktop client to other users, or receives your incoming messages. It can also talk with other daemons and scripts to block spam, viruses or unwanted connections.

When an email is being received by the MTA, it first checks its recipient and decides what to do with it. If the MTA is responsible as the final destination for it, it will deliver it locally to the particular user via the LDA (see next section). If not, it will determine the responsible MTA for that recipient by resolving the domain part of the email address (e.g. for `joe@example.com`, it would be `example.com`) and looks up the "MX entry" for this domain (could be something like `mail.example.com`). Then, the MTA is establishing a connection to this host using the Simple Mail Transport Protocol (SMTP).

Famous MTAs are Sendmail, Exim, Qmail, Postfix, MS Exchange.

### 26.2.2 LDA, IMAP/POP3 Server

If the MTA decides to deliver an email locally, it will send it to a **Local Delivery Agent** (or MDA, **Mail Delivery Agent**). It has the task of delivering the emails into the local mailboxes of the users. Finally, an IMAP server is providing an interface for the user's mail client applications (see next section) for the mailboxes.

Famous LDAs are procmail, mailldrop, Dovecot deliver. Famous IMAP servers are Courier, Cyrus IMAPd, Dovecot.

### 26.2.3 MUA

The **Mail User Agent** is the application a user is running on her desktop to access her emails. This establishes a connection to the mail server via the IMAP or POP3 protocol.

Very often today, emails are retrieved and sent via a web interface. Therefore, a web application is acting as a MUA and is itself running on a web server, providing access to the user's mailbox via a web browser.

Famous MUAs are MS Outlook, Windows Mail, Thunderbird (desktop applications), Squirrel-Mail, RoundCube (webmail servers)

Many other components can be employed within an email server setup. Often, especially with larger installations for more than a few users, a database backend is used for the user accounts. This is referred to as "virtual users", as email users do not have to have a system account to log on to the operating system, but only for the mail server. Also, to centralize the actual authentication users have to go through to send and/or retrieve their emails, an authentication daemon (e.g. Cyrus-SASL or Dovecot-SASL) is used which gives a standardized interface for the agents to verify user data upon login.

External filtering systems can be used for decreasing the amount of unwanted commercial emails a user receives ("spam") or to perform virus-filtering.

## 26.3 Installation

In the following, an example setup will be discussed. In this example, Postfix will be used as MTA, Dovecot as LDA, MySQL as user database backend, and Dovecot-SASL as the user authentication daemon. To run a fully-working mail server which can not only send but also receive emails via the Internet, control over an Internet domain (e.g. "example.com") is necessary for other mail servers being able to connect to this system.

### 26.3.1 Postfix

Postfix was created in 1997 by IBM and released 1998 as "free software". It is developed as a fast, easy-to-administrate and secure MTA. It is the standard MTA on Ubuntu.

Postfix has a particular resilience against buffer overflows and can handle large amounts of e-mail. A Postfix system implements a cooperating network of different daemons, i.e. it consists of many small programs (daemons) which fulfill a very particular task. Each daemon is using minimum privileges so that if a daemon is compromised (hacked), the impact remains limited to that daemon and cannot spread throughout the entire system. Also, this way the system can be extended and modified easily. Only one process has root privileges (master). Most daemons can be easily chrooted and communicate through named pipes or UNIX-domain sockets.

Installing Postfix under Ubuntu or Debian is done by simple typing

```
$ aptitude install postfix
```

If it is not started automatically during the installation, the most basic configuration values for Postfix can be set via

```
$ dpkg-reconfigure postfix
```

Depending on the version used to install, some of these questions might appear and should be answered as follows. These settings can all be adjusted in the configuration files:

```
General type of mail configuration: Internet Site
System mail name: server1.example.com
Root and postmaster mail recipient: <admin_user_name>
Other destinations for mail: server1.example.com, example.com, localhost.
                           example.com, localhost
Force synchronous updates on mail queue?: No
Local networks: 127.0.0.0/8
Mailbox size limit (bytes): 0
Local address extension character: +
Internet protocols to use: all
```

Generate certificates to be used for TLS encryption and/or certificate Authentication:

```
$ touch smtpd.key
$ chmod 600 smtpd.key
$ openssl genrsa 1024 > smtpd.key
$ openssl req -new -key smtpd.key -x509 -days 3650 -out smtpd.crt # has prompts
$ openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -
  days 3650 # has prompts
$ mv smtpd.key /etc/ssl/private/
$ mv smtpd.crt /etc/ssl/certs/
$ mv cakey.pem /etc/ssl/private/
$ mv cacert.pem /etc/ssl/certs/
```

Configure Postfix to do TLS encryption for both incoming and outgoing mail:

```
$ postconf -e 'smtp_tls_security_level = may'
$ postconf -e 'smtpd_tls_security_level = may'
$ postconf -e 'smtpd_tls_auth_only = no'
$ postconf -e 'smtp_tls_note_starttls_offer = yes'
$ postconf -e 'smtpd_tls_key_file = /etc/ssl/private/smtpd.key'
$ postconf -e 'smtpd_tls_cert_file = /etc/ssl/certs/smtpd.crt'
$ postconf -e 'smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem'
$ postconf -e 'smtpd_tls_loglevel = 1'
$ postconf -e 'smtpd_tls_received_header = yes'
$ postconf -e 'smtpd_tls_session_cache_timeout = 3600s'
$ postconf -e 'tls_random_source = dev:/dev/urandom'
$ postconf -e 'myhostname = server1.example.com' # remember to change this to
      yours
```

The `postconf` command does nothing more than changing a value in Postfix' main configuration file `/etc/postfix/main.cf` – it can always also be done by editing this file.

Restart the postfix daemon to activate the changes:

```
$ /etc/init.d/postfix restart
```

Now basic email functionality should be working. Testing can be done by connecting a mail client program (e.g. Thunderbird) to use this Postfix installation as the outgoing (SMTP) server and trying to send an email to an existing email account on the Internet.

### 26.3.2 Dovecot

Installing Dovecot on Ubuntu is as easy as typing:

```
$ aptitude install dovecot-common dovecot-imapd dovecot-pop3d
```

The main configuration needs to be done in the file `/etc/dovecot/dovecot.conf`. The following instructions are changes or additions to this file. It should be treated with caution; if a setting already exists, it has to be changed in place, never just added at the end, because many settings belong to certain environments inside the file.

Which protocols Dovecot should be able to speak with mail clients has to be set:

```
protocols = pop3 pop3s imap imaps
```

Then, set the mailbox format (we chose "maildir" earlier):

```
mail_location = maildir:~/Maildir (for maildir)
```

To configure dovecot to use SSL, amend the following lines (in some cases you may simply have to remove the # symbol from the beginning of the line):

```
ssl_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
ssl_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
ssl_disable = no
```

To particularly support Thunderbird as email client, edit the following:

```
protocol imap {
    ...
login_greeting_capability = yes
imap_client_workarounds = tb-extra-mailbox-sep
}
```

As LDA, the "deliver" program from Dovecot shall be used rather than Postfix putting emails directly into a maildir folder. For this, a line needs to be added to the file /etc/postfix/master.cf

```
dovecot unix - n n - - pipe flags=DRhu user=vmail:mail argv=/usr/lib/dovecot/deliver -d $(recipient)
```

Tell Postfix to actually use the deliver program by changing these values:

```
$ postconf -e 'mailbox_transport = dovecot'
$ postconf -e 'mailbox_command = /usr/lib/dovecot/deliver'
$ postconf -e 'dovecot_destination_recipient_limit = 1'
$ postconf -e 'virtual_transport = dovecot'
```

The permissions of the Dovecot deliver program need to be adjusted:

```
$ chmod 04750 /usr/lib/dovecot/deliver
```

Dovecot will fail to start if isn't configured with a postmaster address. This address will be informed about all deliver problems that might occur. Put it in the "lda" environment inside /etc/dovecot/dovecot.conf

```
protocol lda {  
...  
    postmaster_address = your@email.tld  
}
```

Inside the "imap" environment, the right `imap` and `imap-login` executables need to be set:

```
login_executable = /usr/lib/dovecot/imap-login  
mail_executable = /usr/lib/dovecot/imap
```

### 26.3.3 MySQL

First, install the necessary packages:

```
$ aptitude install postfix-mysql mysql-client mysql-server
```

Then, set a good MySQL root password (not to be confused with the Linux root user):

```
$ sudo mysqladmin -u root password <rootpassword>
```

To create the tables needed for Postfix and the other daemons, the following sql script needs to be inserted into MySQL. For this, create a new file (e.g. `postfixadmin-mysql.sql`) with an editor and paste the script into it:

```
$ vim postfixadmin-mysql.sql
```

This is the script which needs to be copied:

```
#-----Start copy  
-----  
#  
CREATE DATABASE postfix;  
  
GRANT SELECT ON postfix.* TO postfix@localhost IDENTIFIED BY 'postfixpassword';  
GRANT SELECT, INSERT, DELETE, UPDATE ON postfix.* TO postfixadmin@localhost  
    IDENTIFIED BY 'postfixadmin';  
  
USE postfix;  
#  
# Table structure for table admin
```

```
#  
CREATE TABLE admin (  
    username varchar(255) NOT NULL default '',  
    password varchar(255) NOT NULL default '',  
    created datetime NOT NULL default '0000-00-00 00:00:00',  
    modified datetime NOT NULL default '0000-00-00 00:00:00',  
    active tinyint(1) NOT NULL default '1',  
    PRIMARY KEY (username),  
    KEY username (username)  
) COMMENT='Postfix Admin - Virtual Admins';  
  
#  
# Table structure for table alias  
#  
CREATE TABLE alias (  
    address varchar(255) NOT NULL default '',  
    goto text NOT NULL,  
    domain varchar(255) NOT NULL default '',  
    created datetime NOT NULL default '0000-00-00 00:00:00',  
    modified datetime NOT NULL default '0000-00-00 00:00:00',  
    active tinyint(1) NOT NULL default '1',  
    PRIMARY KEY (address),  
    KEY address (address)  
) COMMENT='Postfix Admin - Virtual Aliases';  
  
#  
# Table structure for table domain  
#  
CREATE TABLE domain (  
    domain varchar(255) NOT NULL default '',  
    description varchar(255) NOT NULL default '',  
    aliases int(10) NOT NULL default '0',  
    mailboxes int(10) NOT NULL default '0',  
    maxquota int(10) NOT NULL default '0',  
    transport varchar(255) default NULL,  
    backupmx tinyint(1) NOT NULL default '0',  
    created datetime NOT NULL default '0000-00-00 00:00:00',  
    modified datetime NOT NULL default '0000-00-00 00:00:00',  
    active tinyint(1) NOT NULL default '1',  
    PRIMARY KEY (domain),  
    KEY domain (domain)  
) COMMENT='Postfix Admin - Virtual Domains';  
  
#  
# Table structure for table domain_admins  
#  
CREATE TABLE domain_admins (  
    username varchar(255) NOT NULL default '',  
    domain varchar(255) NOT NULL default '',  
    created datetime NOT NULL default '0000-00-00 00:00:00',  
    active tinyint(1) NOT NULL default '1',  
    KEY username (username)  
) COMMENT='Postfix Admin - Domain Admins';  
  
#  
# Table structure for table log  
#  
CREATE TABLE log (  
    timestamp datetime NOT NULL default '0000-00-00 00:00:00',  
    username varchar(255) NOT NULL default '',
```

```
domain varchar(255) NOT NULL default '',
action varchar(255) NOT NULL default '',
data varchar(255) NOT NULL default '',
KEY timestamp (timestamp)
) COMMENT='Postfix Admin - Log';

#
# Table structure for table mailbox
#
CREATE TABLE mailbox (
    username varchar(255) NOT NULL default '',
    password varchar(255) NOT NULL default '',
    name varchar(255) NOT NULL default '',
    maildir varchar(255) NOT NULL default '',
    quota int(10) NOT NULL default '0',
    domain varchar(255) NOT NULL default '',
    created datetime NOT NULL default '0000-00-00 00:00:00',
    modified datetime NOT NULL default '0000-00-00 00:00:00',
    active tinyint(1) NOT NULL default '1',
    PRIMARY KEY (username),
    KEY username (username)
) COMMENT='Postfix Admin - Virtual Mailboxes';

#
# Table structure for table vacation
#
CREATE TABLE vacation (
    email varchar(255) NOT NULL default '',
    subject varchar(255) NOT NULL default '',
    body text NOT NULL,
    cache text NOT NULL,
    domain varchar(255) NOT NULL default '',
    created datetime NOT NULL default '0000-00-00 00:00:00',
    active tinyint(1) NOT NULL default '1',
    PRIMARY KEY (email),
    KEY email (email)
) COMMENT='Postfix Admin - Virtual Vacation';
#-----End copy
-----
```

Now you need to tell Postfix where the control information is stored in the database. You need to create the following four text files in /etc/postfix for that reason.

```
$ vim /etc/postfix/mysql_virtual_alias_maps.cf
```

Add the following (replace the "postfixpassword" with some password of your choice. But remember to use it from now on for each appearances of this password.):

```
user = postfix
password = postfixpassword
hosts = 127.0.0.1
dbname = postfix
table = alias
```

```
select_field = goto
where_field = address
```

```
$ vim /etc/postfix/mysql_virtual_domains_maps.cf
```

```
user = postfix
password = postfixpassword
hosts = 127.0.0.1
dbname = postfix
table = domain
select_field = domain
where_field = domain
#additional_conditions = and backupmx = '0' and active = '1'
```

```
$ vim /etc/postfix/mysql_virtual_mailbox_maps.cf
```

```
user = postfix
password = postfixpassword
hosts = 127.0.0.1
dbname = postfix
table = mailbox
select_field = maildir
where_field = username
#additional_conditions = and active = '1'
```

```
$ vim /etc/postfix/mysql_virtual_mailbox_limit_maps.cf
```

```
user = postfix
password = postfixpassword
hosts = 127.0.0.1
dbname = postfix
table = mailbox
select_field = quota
where_field = username
#additional_conditions = and active = '1'
```

```
$ vim /etc/postfix/mysql_relay_domains_maps.cf
```

```
user = postfix
password = postfixpassword
hosts = 127.0.0.1
```

```
dbname = postfix
table = domain
select_field = domain
where_field = domain
additional_conditions = and backupmx = '1'
```

Adjust the permissions of these files to protect them from unauthorized access:

```
$ chgrp postfix /etc/postfix/mysql_*.cf
$ chmod 640 /etc/postfix/mysql_*.cf
```

This system can hold mailboxes for thousands of users. All of these users are virtual users and none of them is a Linux system user, hence these users can not store their mail in our system's hard disk. For this reason, a single Linux system user account will be used to map all virtual users to and who will be the owner of all mailboxes. It has to be created:

```
$ groupadd -g 5000 vmail
$ useradd -m -g vmail -u 5000 -d /home/vmail -s /bin/bash vmail
```

Now the mysql files need to be referenced in the main.cf file:

```
$ vim /etc/postfix/main.cf
```

Add the following:

```
# Virtual Mailbox Domain Settings

virtual_alias_maps = mysql:/etc/postfix/mysql_virtual_alias_maps.cf
virtual_mailbox_domains = mysql:/etc/postfix/mysql_virtual_domains_maps.cf
virtual_mailbox_maps = mysql:/etc/postfix/mysql_virtual_mailbox_maps.cf
virtual_mailbox_limit = 51200000
virtual_minimum_uid = 5000
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
virtual_mailbox_base = /var/vmail
virtual_transport = virtual

# Additional for quota support

virtual_create_maildirsize = yes
virtual_mailbox_extended = yes
virtual_mailbox_limit_maps = mysql:/etc/postfix/
    mysql_virtual_mailbox_limit_maps.cf
```

```

virtual_mailbox_limit_override = yes
virtual_maildir_limit_message = Sorry, the your maildir has overdrawn your
    diskquota, please free up some of spaces of your mailbox try again.
virtual_overquota_bounce = yes

```

Finally, check if these settings are correct in /etc/postfix/main.cf

```

myhostname = mail.example.com
mydestination = #Remains blank since we are going to host virtual domains
relayhost = #Remains blank unless you are going to use your ISP's SMTP server
    mail sending out mails. In which case it would be set to the host name of
the ISP's SMTP server

```

#### 26.3.4 Dovecot SASL

First, Dovecot needs to be told to provide SASL client authentication. Therefore, edit /etc/dovecot/dovecot.conf and add the following values in the correct places:

```

auth default {
    mechanisms = plain login
socket listen {
    #master {
        # Master socket provides access to userdb information. It's typically
        # used to give Dovecot's local delivery agent access to userdb so it
        # can find mailbox locations.
        #path = /var/run/dovecot/auth-master
        #mode = 0600
        # Default user/group is the one who started dovecot-auth (root)
        #user =
        #group =
    }
    client {
        # The client socket is generally safe to export to everyone. Typical use
        # is to export it to your SMTP server so it can do SMTP AUTH lookups
        # using it.
        path = /var/spool/postfix/private/auth-client
        mode = 0660
        user = vmail
        group = mail
    }
}
}

```

After that, restart Dovecot:

```
$ /etc/init.d/dovecot restart
```

Postfix needs to be configured to use this service. Either edit main.cf or use these commands:

```
$ postconf -e 'smtpd_sasl_type = dovecot'
$ postconf -e 'smtpd_sasl_path = private/auth-client'
$ postconf -e 'smtpd_sasl_auth_enable = yes'
$ postconf -e 'smtpd_recipient_restrictions = permit_sasl_authenticated,
    permit_mynetworks,reject_unauth_destination'
```

Restart Postfix:

```
$ /etc/init.d/postfix restart
```

### 26.3.5 PostfixAdmin

This package provides a web interface for the database backend (MySQL) for day-to-day administration tasks of the mail server (i.e. managing domains, users, mailboxes etc.). It needs to be downloaded manually from the project homepage (<http://postfixadmin.sourceforge.net/> – be sure to pick the .deb file) and then installed using dpkg, e.g.:

```
$ dpkg -i postfixadmin_2.3.2_all.deb
```

Edit the file `/usr/share/postfixadmin/config.inc.php` and set the following values (according to the values you set earlier):

```
$CONF['database_type'] = 'mysqli';
$CONF['database_host'] = 'localhost';
$CONF['database_user'] = 'postfix';
$CONF['database_password'] = '<postfixpassword>';
$CONF['database_name'] = 'postfix';
$CONF['database_prefix'] = '';
```

For this to work, it needs a running Apache (see 20.2 for installation). You can point your browser to `http://localhost/postfixadmin/setup.php` and follow the instructions on screen. After you entered a password for future authentication at PostfixAdmin, it will be printed on the screen in encrypted form (the web server does not have permissions to write directly into the configuration file) - you have to put this string again into `/usr/share/postfixadmin/config.inc.php`:

```
$CONF['setup_password'] = '<some scrambled string>';
```

When all warnings or errors from the setup page have been fixed, domains and mailboxes can now be added via <http://localhost/postfixadmin>

If anything is not working as expected, the first place to have a look is the mail log file found in `/var/log/mail.log`. All daemons related to the mail service (Postfix, Dovecot etc.) are writing in this file, and mostly in a comprehensive and human-readable form. Often, problems occur due to missing permissions of files or folders e.g. or wrong user mappings.

# Samba

---

## Contents

<b>27.1 SMB/CIFS . . . . .</b>	<b>231</b>
<b>27.2 An Introduction to Samba . . . . .</b>	<b>232</b>
<b>27.3 Installation of Samba . . . . .</b>	<b>232</b>
<b>27.4 The Samba Server Configuration . . . . .</b>	<b>233</b>
27.4.1 Configuration File Structure . . . . .	234
27.4.2 Most important configuration options . . . . .	235
27.4.3 Testing a configuration . . . . .	238
27.4.4 Creating users . . . . .	238
<b>27.5 Connecting to Samba Shares . . . . .</b>	<b>238</b>
<b>27.6 Printer Shares . . . . .</b>	<b>241</b>
<b>27.7 Summary of Samba Daemon and common used Commands . . . . .</b>	<b>242</b>

To be a good Samba administrator, you need to know basic Unix/Linux system and network administration and have a good understanding of Windows File System and networking fundamentals. In this chapter, we want to learn how Samba fills in the "gray area" between Unix and Windows. Once you know how everything fits together, you should able to configure a Samba server to provide your network with high-performance resources.

The official documentation <sup>1</sup> (and in particular the man pages) given on this web page are taken from the latest development version of Samba. If you are using an earlier version of Samba then you may find some differences.

## 27.1 SMB/CIFS

The **Server Message Protocol (SMB)** operates as an application-layer network protocol (see “OSI model” of networking). It is used mainly to provide shared access to files, printers, serial interfaces and miscellaneous communications between nodes on a network. Since most usage of SMB involves computers running Microsoft Windows, it is known as Microsoft Windows Network in the Windows World.

The **Common Internet File System (CIFS)** is a extension of the SMB protocol that provides additional services like Windows-RPC-services (remote procedure calls) and NT-domain-services. Usually it is used in preference to SMB.

---

<sup>1</sup><http://samba.org/samba/docs/>

## 27.2 An Introduction to Samba

**Definition** To describe Samba, we begin with the explanation of things that it can do for us. Samba implements the CIFS network protocol. By supporting this protocol, Samba enables computers running Unix-based operating systems to communicate with Microsoft Windows and other CIFS-enabled clients and servers. Some examples of common services offered by Samba are:

- Samba can be configured as a server for share folders and printers, including PDF pseudo-printers so all the computers in your network may write PDF files
- Act as a domain controller in a Windows network (authenticating users, etc.)
- Do some more complex things, such as using a Windows domain controller to authenticate the users of a Linux/UNIX machine

Samba is a popular open source software that allows end users to access and use files, printers, and other commonly shared resources on a local network or on the Internet. Samba is often referred to as a network file system.

Samba supports the common client/server protocol of Server Message Block (SMB) and Common Internet File System (CIFS). Using client software that also supports SMB/CIFS (for example, most Microsoft Windows products), an end user sends a series of client requests to the Samba server on another computer in order to open that computer's files, access a shared printer, or access other resources. The Samba server on the other computer responds to each client request, either granting or denying access to its shared files and resources.

**Main Daemons in Samba 3.0** Samba's current stable release, revolves around three Unix daemons:

- smbd: this daemon handles file and printer sharing and provides authentication and authorization for SMB clients
- nmbd: this daemon handles Samba's NetBIOS name registration, implements a Microsoft-compatible NetBIOS Name Server service, also referred to a WINS server.
- winbindd: this daemon communicated with domain controllers for providing information such as the groups to which a user belongs.

## 27.3 Installation of Samba

**Overview** A typical Samba installation with compiling from source takes about one hour to complete, including downloading the source code and compiling them, setting up the configuration files, and testing the server. On Debian there are a few packages that contains all necessary components to turn a Debian server into a powerful file and printer server. The main package is `samba`. It depends on `samba-daemon`, that provides the files server that are used by both server

and SMB clients. It is simple to install Samba on a Ubuntu machine by using the following command:

```
$ sudo aptitude install samba smbfs  
// backup the original configuration file  
$ sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.original
```

The installed package contains two main services of the Samba suite: `nmbd` and `smbd`. The man-page of `nmbd` says:

*The nmbd is a server that understands and can reply to NetBIOS over IP name service requests, like those produced by SMB/CIFS [...] clients. SMB/CIFS clients, when they start up, may wish to locate an SMB/CIFS server. That is, they wish to know what IP number a specified host is using.*

The man-page of `smbd` says:

*smbd is the server daemon that provides file sharing and printing services to the [...] clients. The server provides file space and printer services to clients using the SMB (or CIFS) protocol.*

As well, the standard packages contain other tools like `smbpasswd`, `testparm`, that will be used during the configuration process.

**Starting/Stopping Samba daemon** The Samba service is started, stopped, restarted or an reloading of the configuration is invoked by:

```
$ sudo start smbd      // {start, stop, restart, reload}
```

As well, the `nmbd` service is started, stopped, restarted by using:

```
$ sudo start nmbd      // {start, stop, restart}
```

We can check if the services are started like this:

```
$ ps ax | grep mbd
```

## 27.4 The Samba Server Configuration

The Samba server configuration files are located in `/etc/samba`. The main configuration file is called `smb.conf`, which uses the same format as Windows `.ini` file.

### 27.4.1 Configuration File Structure

An overview of Samba configuration file from a high level:

```
[global]
...
[homes]
...
[printers]
...
[name_of_share]
...
```

**[global]** Server-wide settings are defined here, and any options that apply to shares will be used as a default in all share definitions, unless overridden within the share definition.

**[homes]** If a client attempts to connect to a share that doesn't appear in the smb.conf file, Samba will search for a **[homes]** share in the configuration file. If a **[homes]** share exists, the unresolved share name is assumed to be a Unix username. If that username appears in the password database on the Samba server, Samba assumes the client is a Unix user trying to connect to her home directory on the server.

**[printers]** If the **printers** share is defined, look up the requested share name in the **printcap** file. If a match is found, use the shared printer as the service in response to the client's request.

**[print\$]** This share provides access to the server's repository of print drivers needed to support point-and-print functionality. All authenticated users are able to copy files from this share, but only administrator accounts can add or modify files. The administrator can manage the printer drivers centrally by using this share.

**[name\_of\_share]** For each shared resource, you can define a share with any proper name as individual section. When Samba server receives a request to connect to a share, if a match is found, authorize this request against this share.

Listing 27.1: Example of Samba Configuration

```
[global]
workgroup = BREAKFAST
server string = %h server (Samba, Ubuntu)
map to guest = Bad User
security = share
syslog = 0
log file = /var/log/samba/log.%m

[printers]
comment = All Printers
```

```

path = /var/spool/samba
create mask = 0700
printable = Yes
browseable = No

[print$]
comment = A share to store all Printer Drivers
path = /var/lib/samba/printers

[public]
comment = Shared Folder
path = /var/tmp
public = yes
writable = yes
create mask = 0700
directory mask = 0700
force user = nobody
force group = nogroup
available = yes
browseable = yes

```

### 27.4.2 Most important configuration options

Since Samba has a huge variety of configuration possibilities we can only cover a few important configuration keywords that should enable the reader to configure Samba as used in a common scenario like a small local network.

To learn all the features you should read the manpage<sup>2</sup>:

```
$ man smb.conf
```

Note: these are only those parameters that are regarded as the most important. The notation of the parameters in the config file is *parameter = value*.

#### Server Naming options

- **workgroup** - name the workgroup or the NT-domain the server should be part of
- **server string** - sets what will appear next to the machine name in the browse list
- **netbios name** - the name you will see in ‘Network neighborhood, defaults to the hostname

**Printing Options** If a [printers] section occurs users are able connect to any printer specified in the local host’s printcap file. This section must contain **printable=yes**, if specified otherwise the configuration will be refused to load.

<sup>2</sup> <http://www.samba.org/samba/docs/man/manpages-3/smb.conf.5.html>

- **printcap name** - override the default printcap name (usually */etc/printcap*)
- **load printers** - load all the printers in the printcap for browsing, default is yes
- **printing** - specify the print system type

### Logging options

- **log file** - specify which log file(s) to use
- **max log size** - limits the size of the log files (in KB)
- **log level** - set the log (verbosity) level (0-10)

### Security and Domain Membership options

- **security** - one of the most important settings, option affects how clients respond to samba, default is security=user where clients must first “log-on” with a valid username and password
- **encrypt password** - encryption of the user passwords, default is yes
- **hosts allow** - allows you to restrict connections to machines on your local network
- **guest account** - use if a guest account is wanted
- **include** - include extern (configuration) files
- **interfaces** - configure Samba to use multiple interfaces, e.g. several subnets
- **local master** - If set to yes the nmbd tries to become the master browser in the subnet, default is yes; this is no guarantee that nmbd will become the local master browser, but that will participate in the elections. If set to no it will never be a local master.
- **OS level** - determines the precedence of the server in the master browser selection. The higher that value the more likely is it that the server wins the election. You can use values from 0-255, default is 20. Use this reasonably.
- **domain master** - specifies Samba to be the Domain Master Browser. This allows Samba to collate browse lists between subnets.

### Domain Control Options

- **domain logons** - enable this if you want Samba to be a Primary Domain Controller for Windows workstations
- **logon script** - if you enable domain logons then you may want a per-machine or per user logon script
- **logon path** - specify where to store roaming profiles

- **passdb backend** - this option allows the administrator to chose which backend will be used for storing user and possibly group information. The possible backends are either a simple file *smbpasswd*, via TDB (*tdbsam*) or via an LDAP (*ldapsam*). The default is *smbpasswd* with the file */etc/samba/smbpasswd*.  
For LDAP based passdb backend take a further look at the *ldap* and *idmap* related parameters in the man page.
- manage samba users, groups, machines in the domain with parameters *s.a. add user script*, *delete user script*, *add group script*, .... For a LDAP based user management there are special scripts installed via the package *smbldap-tools*.

**Name Resolution Options** All NetBIOS names must be resolved to IP addresses. There are a few options to configure how the name resolution is done.

- **name resolv order** - specify the order in which the name resolution is realized. The default order is "host lmhosts wins broadcast".
- **wins support** - either yes or no, enables support for wins (windows INTERNET name service), only one computer in a network should act as wins server
- **wins server** - define the WINS server, you can define a **wins proxy** or a **dns proxy** as well.

#### Options in the shares section (includes [printers], [homes])

- **comment** - text field seen next to the share when querying the server to list what shares are available
- **path** - specify a directory to which the user of the service is to be given access
- **browsable** - controls whether this share is seen in the list of available shares in the browse list; default is yes
- **guest OK** - if set to yes, no password is required to connect to the share, the privileges will be those of the **guest account** (specified in the global section); default is no; **public** is an alias to this option
- **read only** - if set to yes, user may not create or modify files in the service's directory, default is yes; this is the inverted synonym to **writable**
- **guest only** - if set to yes only guest connections to this service are permitted; will only take effect if guest OK is set
- **printable** - if set to yes, clients may write to and submit to the printing spool
- For more detailed rights management on a share read the documentation for at least the following parameters: **valid users**, **write list**, **force group**, **force user**, **create mask**, **directory mask**, **force create mode**, **force directory mode**

### 27.4.3 Testing a configuration

After having written the *smb.conf* one should run the following command to check for errors in the configuration:

```
$ testparm {config file}
```

If there is no config file supplied testparm will look for the standard configuration file */etc/samba/smb.conf* and will check it. It returns error code of 1 if it finds errors in the configuration file, else it returns 0. If called directly testparm dumps the configuration to standard output.

### 27.4.4 Creating users

Assuming that we use a the smbpasswd backend for storing the user information, in order to enable the user "myuser" with the password "mypassword" to use Samba, the following things must exist:

1. A valid *myuser* Unix account. *myuser* does not need to be able to log in, and *myuser*'s Unix password is not used by Samba, so you can set to a dummy value if you like. If *myuser* also logs in interactively to your Samba server, that's OK, too.
2. A valid *myuser* entry in the smbpasswd file. *myuser* can be added to the smbpasswd file by using the smbpasswd command and by typing the following:

```
$ sudo smbpasswd -a sambauser
```

Repeat this procedure for every new Samba user.

## 27.5 Connecting to Samba Shares

Assume that we want to connect the Windows clients to the shares on a Samba server and the Windows clients are located in a workgroup with name **BREAKFAST** in a local network, as shown in Figure 27.1.

**Windows Setup** Before a Windows client can connect to a Samba server or other any CIFS server, a few components must be configured. At a minimum, we need the following software components, as shown in Figure 27.2.

- TCP/IP network protocol

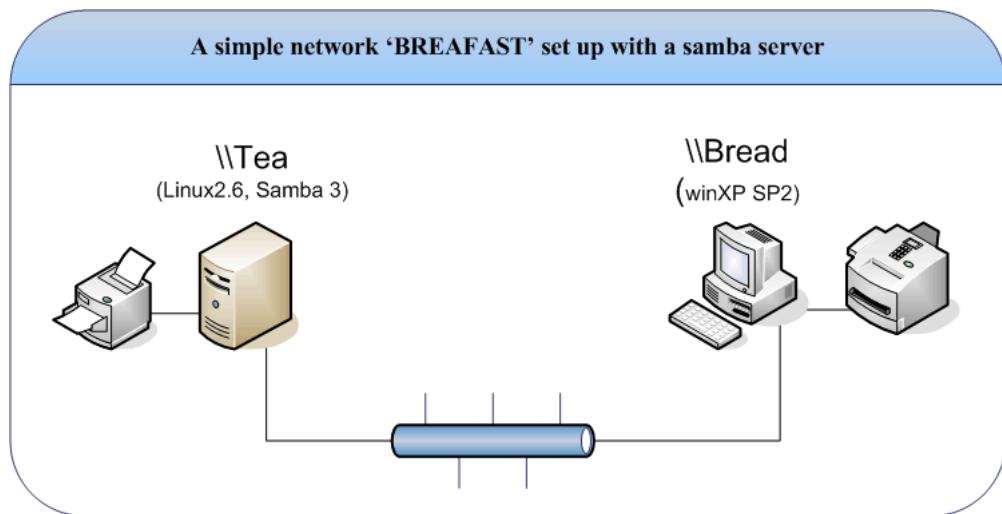


Figure 27.1: A simple network with Samba Server

- Client for Microsoft Networks component
- File and Printer Sharing for Microsoft Networks

**Configuring TCP/IP** Open the Internet Protocol(TCP/IP) Properties dialog box(Figure 27.3). You have to configure the settings for IP address, WINS Address, and DNS server properly, click OK in each open dialog box and close them to complete the configuration. A detailed guidance will be presented in the slides during the course.

**Connecting to the Samba Server** Assume that we have created a Linux user named `sambauser` by running `sudo smbpasswd -a sambauser` on the Samba server. Now we might be able to map to the share `[public]` that is located on the Samba server `Tea`.

```
c:\> net use t: \\tea\public /user:sambauser password
```

If the username and password is valid for the share on the server, you should be greeted with the following line:

```
The command completed successfully.
```

Now a new device name T: should be added to the Windows system. You are able to access the shared files on this service.

**Browsing the Samba Server** Alternatively you can drill down through the entire network to this workgroup `BREAKFAST` and expand the list of servers and clients, as shown in Figure 27.4.

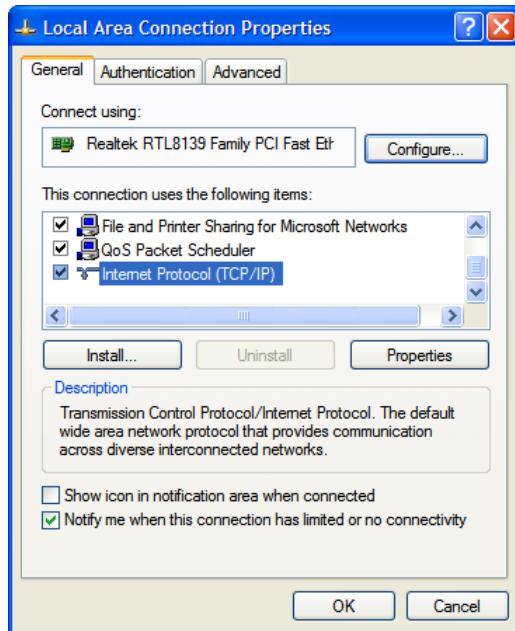


Figure 27.2: List of installed network components for local area connection

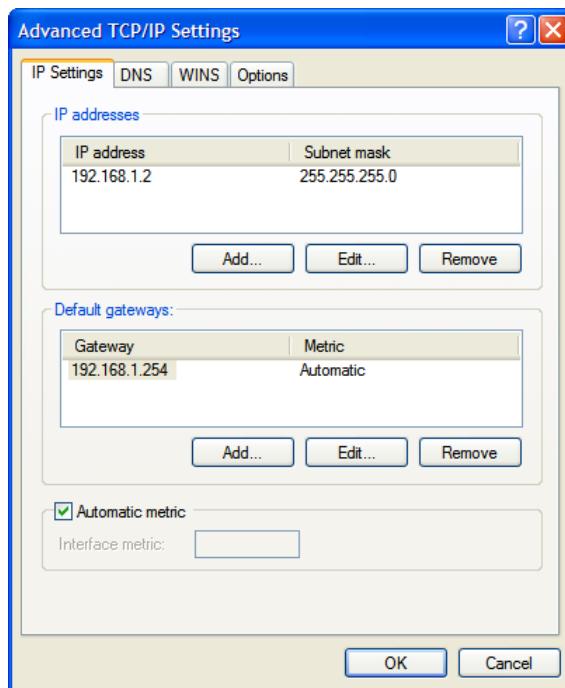


Figure 27.3: IP settings of the advanced TCP/IP settings dialog box

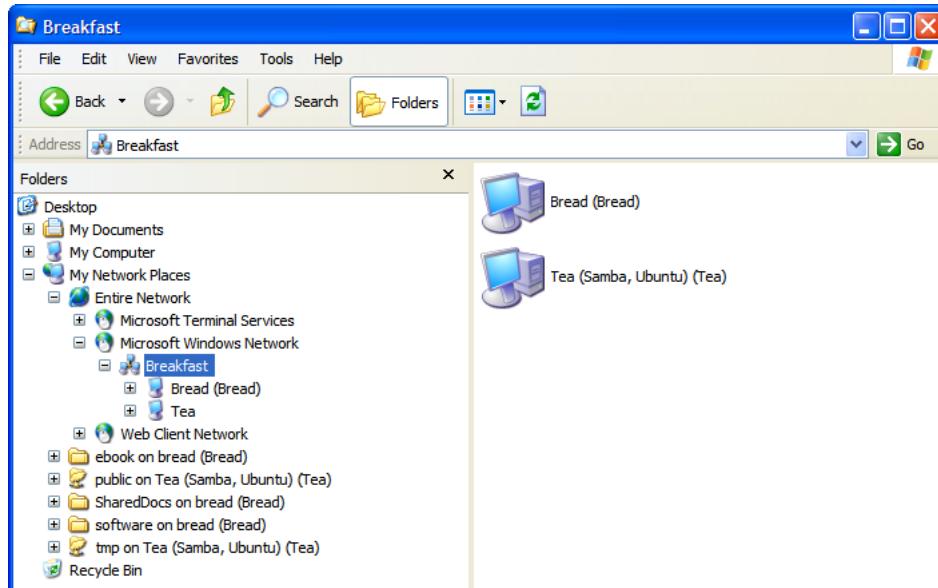


Figure 27.4: Viewing My network places through the windows explorer

By selecting one of machine in the network, we can view the available shares like File Sharing or Printer Sharing.

## 27.6 Printer Shares

**CUPS** Common Unix Printing System(CUPS) allows a computer to act as a print server that can accept print jobs from clients, process them, and send them to the appropriate printer. CUPS is becoming more and more common in the networks and it is integrated in Samba as `cupsd`. To verify the printer daemon in Samba, we can look for `HAVE_CUPS` in the build options output from `smbd`:

```
$ smbd -b | grep cups
```

**Installing CUPS and PDF printer** If CUPS is not installed as default on your machine, install the systems by using the following command:

```
$ sudo aptitude install cups cups-pdf
```

**CUPS on Samba** To share the printers on Samba server by using CUPS printing system, we should add the following lines to the `[global]` section and update `[printers]` section:

```
#### smb.conf
[global]
...
printing = cups
printcap name = /etc/printcap
load printers = yes

[printers]
comment = all printers on Samba Server
path = /var/spool/samba
printable = yes
read only = yes
create mask = 0700
public = yes
```

After reloading Samba configuration, we should be able to view the shared printers on Samba Server from the Windows Network.

## 27.7 Summary of Samba Daemon and common used Commands

**SMB URI Syntax** The SMB URI Syntax is similar to the URIs used to access resources on the Internet, such as *ftp://ftp.ziik.org*.

The following examples are commonly used to connect a share on the Samba server:

```
smb://workgroup
smb://server
smb://server/share
smb://workgroup/server/share
smb://username:password@server/share
smb://username:password@workgroup/server/share
```

**findsmb** This command reports information about systems on the subnet that respond to SMB name-query requests. The report includes the IP address, NetBIOS name, workgroup/domain, and operating system for each system.

```
$ findsmb -B -D
```

**smbclient** This program is very similar to that of FTP. It offers a set of functionalities, such as interactive file transfer, sending message on the SMB network ...

```
$ smbclient -L //server/shared  
$ smbclient //server/shared -U username
```

**smbpasswd** The **smbpasswd** provides the general function of managing encrypted passwords. It must be run by the superuser or an ordinary user. When run by ordinary users, **smbpasswd** can be used only to update their encrypted passwords using the SMB/CIFS password change mechanisms.

```
$ smbpasswd -a username          #add a user to the encrypted password file  
$ smbpasswd -x username          #delete a user from encrypted password file
```

**smbtree** The **smbtree** is similar to the **findsmb**, in that it enumerates workgroup contents.

```
$ smbtree -N
```

**testparm** The **testparm** validates a Samba configuration file for errors. workgroup contents.

```
$ testparm                      #use default conf file  
$ testparm -s smb.conf            #use this conf file  
$ testparm -v                     #list all the parameters, even default values
```



# Primary Domain Controller

---

## Contents

<b>28.1 What is an authentication server . . . . .</b>	<b>245</b>
<b>28.2 What is a LDAP server . . . . .</b>	<b>246</b>
<b>28.3 LDAP Server Setup - step by step . . . . .</b>	<b>246</b>
28.3.1 Hostname and Domain Membership . . . . .	246
28.3.2 Installation of LDAP server . . . . .	247
28.3.3 Testing LDAP . . . . .	250
28.3.4 Installation of MMC . . . . .	251
28.3.5 phpLDAPadmin . . . . .	255
<b>28.4 Samba/Windows Users . . . . .</b>	<b>255</b>
<b>28.5 LDAP-authentication and authorization (NSS-LDAP and PAM-LDAP) . . . . .</b>	<b>261</b>
<b>28.6 Mail Users . . . . .</b>	<b>264</b>
<b>28.7 DNS and DHCP Server . . . . .</b>	<b>265</b>
<b>28.8 Linux Clients . . . . .</b>	<b>270</b>
28.8.1 Configure LDAP connections . . . . .	270
28.8.2 Configure PAM . . . . .	272
28.8.3 Setup pam_mount . . . . .	273
<b>28.9 Windows Clients . . . . .</b>	<b>274</b>

---

In this chapter a step by step tutorial is given to install a LDAP based authentication and authorization server. This server can then be used as a Primary Domain Controller in a Windows Domain as well as a authentication and file server for a network of Linux clients. This allows for homogeneous authentication and file access in heterogeneous networks. It will also act as DHCP and DNS server. The user data in the LDAP database may also be used for authentication of other services like email servers or web applications.

## 28.1 What is an authentication server

The purpose of an “auth”-server is to provide both: authentication and authorization to email users as well as UNIX and Windows users. The user databases of file services (Samba) and the mail services (Postfix, Dovecot), that are supposed to run on a different server, are maintained centrally on the “auth”-server.

For maintaining and administrating the user accounts, running a HTTP proxy (SQUID), a mail server, DHCP and DNS servers the "Mandriva Directory Server" (MDS)<sup>1</sup> is deployed. It consists of a daemon with plugins and a web frontend that makes the administration easier. The configuration is stored in a LDAP directory.

## 28.2 What is a LDAP server

The Lightweight Directory Access Protocol (LDAP) is a protocol for querying and modifying the data of a directory<sup>2</sup>. LDAP databases organize data in trees. A single LDAP database consists of one tree. In this tree, all nodes can have attributes and each node can be the root of a subtree. The names of the nodes is a unique identifier: its “Distinguished Name” (DN). The DN consists of its Relative Distinguished Name (RDN), constructed from some attribute(s) in the entry, followed by the parent entry’s DN. An attribute has a name and one or more values. The attributes are defined in a schema (definition of classes and attributes that define what data can and must be stored in a node).

A LDAP server is typically accessed via network. Connections can optionally be encrypted with SSL/TLS. The standard port is 389 and the port for encrypted connections is 636.

OpenLDAP<sup>3</sup> is an open source implementation of LDAP.

## 28.3 LDAP Server Setup - step by step

Note: the installation is carried out on a Debian GNU/Linux system, but is similarly working on other Linux systems. The domain names and IPs are chosen arbitrarily. The domain name used for the LDAP installation is: ph-uni.de.

### 28.3.1 Hostname and Domain Membership

First of all it is important to have the correct DNS settings (hostname and domain). We will use auth as hostname and itcp.ph-uni.de as domain name.

```
root@auth:~$ echo auth > /etc/hostname
root@auth:~$ nano /etc/hosts
```

```
[...]
192.168.200.253 auth.itcp.ph-uni.de auth
[...]
```

---

<sup>1</sup>A detailed documentation of MDS can be founded here: <http://mds.mandriva.org>

<sup>2</sup>Here, “directory” refers to a database which is heavily optimized for reading, under the assumption that data updates are very rare compared to data reads. A directory is a set of objects with attributes organized logically in a hierarchical manner.

<sup>3</sup>Learn more about OpenLDAP: <http://openldap.org>

```
root@auth:~$ /etc/init.d/hostname.sh start
root@auth:~$ hostname
> auth
root@auth:~$ hostname -f
> auth.itcp.ph-uni.de
root@auth:~$ hostname -s
> auth
```

### 28.3.2 Installation of LDAP server

Basis of the authentication server is a working LDAP server.

```
root@auth:~$ aptitude install ldap-server ldap-utils db4.2-util
root@auth:~$ /etc/init.d/slapd stop
```

For the basic configuration of OpenLDAP, we want to allow unencrypted connections (port 389) to the LDAP daemon only from localhost, and allow encrypted connections on all interfaces from everywhere (port 636).

```
root@auth:~$ nano /etc/default/slapd
```

```
[...]
SLAPD_SERVICES="ldap://127.0.0.1/ ldaps:///      # this will open port \code{389} on localhost and port \code{636} on all interfaces
[...]
```

In order to change the LDAP's root-DN (*Distinguished Name*<sup>4</sup>), we delete the existing database at first.

```
root@auth:~$ rm /var/lib/ldap/*
```

After that the LDAP server has to be reconfigured, we use the package management tool `dpkg` for that purpose.

```
root@auth:~$ dpkg-reconfigure slapd
```

<sup>4</sup>The name/address/path of a node in the LDAP tree.

We have to provide a little information here, concerning the LDAP root, the database backend and other information (adjust this to your needs):

- omit Openldap configuration: no
- domain (2x) : ph-uni.de
- Organization name: Phantasy University
- give a challenging password<sup>5</sup>
- database backend to use: hdb
- purge old db: yes
- sometimes it asks “Move old database?”: Yes
- Allow LDAPv2 protocol?: no

We have not yet finished the configuration at this point. We have to make sure that there are valid and up to date SSL respective TLS certificates installed and if tell LDAP where they reside. Unless there are valid certificates, the LDAP daemon won't start.

### Syslog configuration for OpenLDAP

We are going to configure syslog to not use `/var/log/syslog` to log the LDAP output to. Instead we want OpenLDAP's output to go to `/var/log/ldap.log`.

To begin, we install an extension of the `syslogd` with content-based filtering capabilities and flexible configuration options: `syslog-nginx`, whose configuration file is to be edited.

```
root@auth:~$ aptitude install syslog-nginx
root@auth:~$ nano /etc/syslog-nginx/syslog-nginx.conf
```

```
[..]
# put this before the first 'filter' statement in the configuration file
#####
## ldap.log

filter f_ldap { program("slapd"); };
destination d_ldap { file("/var/log/ldap.log"); };
log {
    source(s_all);
    filter(f_ldap);
    destination(d_ldap);
    flags(final);
};

#####
[..]
```

---

<sup>5</sup>Challenging passwords can be generated using the `apg` tool.

Once the syslog configuration is changed, the syslog-ng daemon has to be restarted:

```
root@auth:~$ /etc/init.d/syslog-ng restart
```

We are still not done laying the foundations for a working LDAP server, we still need valid SSL/TLS certificates.

### SSL/TLS certificates

Since we can not cover this topic here. Please read the following HOWTO <http://www.tc.umn.edu/~brams006/selfsign.html> and take a look at the OpenSSL documentation (<http://www.openssl.org/docs/HOWTO/>)

### Configuring OpenLDAP

In case the LDAP-server is still running (yet not configured), stop it and edit the main configuration file:

```
root@auth:~$ /etc/init.d/slapd stop
root@auth:~$ nano /etc/ldap/slapd.conf
```

```
[...]
loglevel 256          # Set this to 0, if everything is working, as this loglevel
sizelimit 5000         # produces so much output, that it will fill your hard disk
tool-threads 2          # and it makes the LDAP server slower.
dbconfig set_cachesize 0 8388608 0  # cachesize for the BerkleyDB

# put here your respective ssl certificates
TLSCACertificateFile /etc/ssl/ph-uni.de/demoCA/cacert.pem
TLS CertificateFile /etc/ssl/ph-uni.de/auth.ph-uni.de_cert.pem
TLS CertificateKeyFile /etc/ssl/ph-uni.de/auth.ph-uni.de_key.pem
[...]
```

In order to improve the performance of the BerkleyDB we add to its configuration:

```
root@auth:~$ nano /var/lib/ldap/DB_CONFIG
```

```
set_cachesize 0 8388608 0
set_lg_bsize 2097152
[...]
```

After that the database has to be restored to a consistent state to let the changes take effect:

```
root@auth:~$ db4.2_recover -eh /var/lib/ldap/
```

As the files were modified by the root user, we have to reset the permissions so that the LDAP daemon can write to it again:

```
root@auth:~$ chown openldap:openldap /var/lib/ldap/*
```

Client software (like the samba daemon) uses */etc/ldap/ldap.conf* to find the local LDAP server:

```
root@auth:~$ nano /etc/ldap/ldap.conf
```

```
[..]
BASE      dc=ph-uni,dc=de
URI       ldap://127.0.0.1/
TLS_REQCERT never          # this is very important, as OpenLDAP will not
                           # accept any self-signed keys without it
[..]
```

### 28.3.3 Testing LDAP

The LDAP server should be tested thoroughly before continuing. This can be done with the command `ldapsearch`:

```
root@auth:~$ ldapsearch -x -W -H ldap://localhost -D cn=admin,dc=ph-uni,dc=de \
-b dc=ph-uni,dc=de

-x use simple (password) authentication
-W ask for the password
-H URI to contact the LDAP server. Use \code{ldap://host\_or\_IP:port} for
unencrypted network connections, \code{ldaps://host\_or\_IP:port} for
encrypted network connections or \code{ldapi://path\_to\_socket} for local
file base socket connections.
-D binddn (dn for a node) to use for authenticating to the LDAP server (like a
username)
-b start the search at this place in the tree
```

Assuming that at this point only the root node is created the output would be:

```

Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=ph-uni,dc=de> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# ph-uni.de
dn: dc=ph-uni,dc=de
objectClass: top
objectClass: dcObject
objectClass: organization
o: ph-uni.de
dc: ph-uni

# admin, ph-uni.de
dn: cn=admin,dc=ph-uni,dc=de
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: abcdefghijklmnopqrstuvwxyz=

```

Those are the root node (`dc=ph-uni,dc=de`) and the node for the administrator user (`cn=admin,dc=ph-uni,dc=de`).

Furthermore one should look whether the LDAP daemon binds to the right ports and interfaces (port 389 for unencrypted, port 636 for encrypted connection). We use `lsof` here:

```
root@auth:~$ lsof -i -P
```

The output should than be:

```

COMMAND      PID      USER      FD  TYPE DEVICE SIZE NODE NAME
# unencrypted connection only from/to localhost
slapd      1373  openldap  6u  IPv4  3260    TCP  localhost:389 (LISTEN)
# encrypted from everywhere
slapd      1373  openldap  8u  IPv4  3282    TCP  *:636 (LISTEN)
# SSH daemon on port 22 (we need to administer remotely)
sshd      1396      root   3u  IPv6  3384    TCP  *:22 (LISTEN)

```

#### 28.3.4 Installation of MMC

To install the MDS (see above) we need a special package source for our package management system. For this purpose we add the following lines to `/etc/apt/sources.list` (the first two lines

should be already there, if not make sure to have them, since they are important for security reasons):

```
deb      http://ftp.de.debian.org/debian/ lenny main contrib non-free
deb      http://security.debian.org/ stable/updates lenny main contrib non-free
deb      http://mds.mandriva.org/pub/mds/debian lenny main
```

To actually get the new package lists we have to update the apt package database. After that we can install the MMC. It has a web interface which is dependent on Apache web server with PHP and a couple of other packages. Since the dependencies are resolved automatically the following command should install all necessary packages:

```
root@auth:~$ aptitude update
root@auth:~$ aptitude install mmc-web-base mmc-agent python-mmc-plugins-tools
python-mmc-base
```

Some more configuration is needed in order to get MMC running: the LDAP needs the MMC base schema. Schemas define the structure (i.e. structure and syntax of attributes and objectclasses) of entries in LDAP subtrees. The schema is included in */etc/ldap/slapd.conf*.

```
root@auth:~$ cp /usr/share/doc/python-mmc-base/contrib/ldap/mmc.schema /etc/
ldap/schema
root@auth:~$ nano /etc/ldap/slapd.conf
```

```
# Schema and objectClass definitions
[...] # other schemas before mmc.schema
include        /etc/ldap/schema/mmc.schema
[...]
```

To get the changes working we need to restart the LDAP daemon:

```
root@auth:~$ /etc/init.d/slapd restart
```

## Apache with SSL

Before actually being able to work with our MMC web interface we need the Apache running with SSL encryption, as we're going to send passwords over the network: Make sure again that you have valid certificates and keys.

The Listen directive tells Apache on which ports it should bind (see 20.2 on page 140 to recall general Apache configuration):

```
root@auth:~$ nano /etc/apache2/ports.conf
```

```
Listen 443
```

Now we need to tell Apache to use encryption in its (virtual) host configuration:

```
root@auth:~$ nano /etc/apache2/sites-available/default
```

```
<VirtualHost *:443>
[...]
SSLEngine on
SSLCertificateFile /etc/ssl/ph-uni.de/mail.ph-uni.de_cert.pem
SSLCertificateKeyFile /etc/ssl/ph-uni.de/mail.ph-uni.de_key.pem

SSLProtocol all
SSLCipherSuite HIGH:MEDIUM
[...]
```

The SSL-module for apache needs to be enabled:

```
root@auth:~$ a2enmod ssl
```

Now Apache can be restarted:

```
root@auth:~$ /etc/init.d/apache2 restart
```

You should be able to contact the web server over HTTPS now.

Maybe you want to restrict which hosts/IPs can connect to the MMC:

```
root@auth:~$ nano /etc/mmc/apache.conf
```

```
### Add an alias /mmc on www server root
Alias /mmc /usr/share/mmc

### Allow access to mmc web directory to everyone
```

```
<Directory /usr/share/mmc>
    AllowOverride None
    Order allow,deny
#
#           allow from all
    allow from xxx.xxx.xxx.xxx          # put the respective host-
#           IPs here
    php_flag short_open_tag on
</Directory>
```

This configuration file will be read by Apache, because it should be included into Apache's configuration in */etc/apache2/httpd.conf*.

### MMC configuration

Now the actual configuration for MMC needs to be done:

- Change password in */etc/mmc/mmc.ini*
- Set the same password in */etc/mmc/agent/config.ini*
- Set baseDN and password in: */etc/mmc/plugins/base.ini*

If we now start the mmc-agent we should get output like the following:

```
root@auth:~$ /etc/init.d/mmc-agent start
Starting Mandriva Management Console XML-RPC Agent:
[...]
mmc-agent starting...
Created OU ou=Users,dc=ph-uni,dc=de
Created OU ou=Computers,dc=ph-uni,dc=de
Created OU ou=Groups,dc=ph-uni,dc=de
Created OU ou=System,dc=ph-uni,dc=de
Plugin base loaded, API version: 5:0:1 build(572)
done.
```

The Users, Computers, Groups and System subtree should have been created and the base plugin loaded. If so, we have done everything correctly.

Invoking a LDAP search again the LDAP server should return the newly created nodes:

```
root@auth:~$ ldapsearch -x -W -h localhost -D cn=admin,dc=ph-uni,dc=de -b dc=ph
-uni,dc=de
```

Now we can point our web browser to the MMC web interface at <https://auth.itcp.ph-uni.de/mmc/>. For logging we can use the "root" as username and the LDAP admin user (*n=admin,dc=ph-uni,dc=de*) password.

### 28.3.5 phpLDAPadmin

Another useful tool to search within the LDAP and administrate it (which should not be the case very often) is the PHP based webinterface `phpLDAPadmin`. Its available as Debian package or on the project's website<sup>6</sup>.

```
root@auth:~$ aptitude install phpldapadmin
```

Find the following lines in the configuration file `/etc/phpldapadmin/config.php` and make sure they contain the denoted values: we want the web application to connect to ldap at localhost without using TLS (Since it can cause problems and we don't need it here, we don't use encryption. In other cases where phpldapadmin needs to connect to a remote server you should make sure to use encryption!).

```
root@auth:~$ nano /etc/phpldapadmin/config.php
```

```
[..]
$ldapservers->SetValue($i,'server','port','389');

$ldapservers->SetValue($i,'server','host','ldap://localhost');

$ldapservers->SetValue($i,'server','tls',false);
[..]
```

Thus we are able to see the web page in the browser at: <https://auth.itcp.ph-uni.de/phpldapadmin>. The login would then be `cn=admin,dc=ph-uni,dc=de` with the LDAP administration password.

By now we have a running authentication server (based on LDAP) for UNIX users. In order to manage mail and Windows users as well we need to install the respective MMC components.

## 28.4 Samba/Windows Users

Some users will need access to computers running a Microsoft Windows operating system. We want them to be able to log into using the same user and password as for their UNIX and email accounts. The MMC component `python-mmc-samba` will add the Samba management feature to the daemon, the component `mmc-web-samba` will add the corresponding management facilities to the MMC web site.

```
root@auth:~$ aptitude install samba
```

<sup>6</sup><http://phpldapadmin.sourceforge.net/>

You will be asked for a workgroup, type the name you want your domain to have (ITCP maybe). When asked for a WINS setting, type “No”.

```
root@auth:~$ /etc/init.d/samba stop
```

Since we are not concentrating on file services here and for security reasons – and authentication servers should have really tight security – we disable all shares in the Samba configuration, and make it absolutely minimal. This also means – sadly – that we cannot configure the Samba shares with the MMC (we assume the file services to be managed by a different machine).

```
[global]
workgroup = ITCP
security = USER
log file = /var/log/samba/log.%m
max log size = 1000
log level = 3
syslog = 0
invalid users = root
# null passwords = yes
# unix charset = ISO8859-1
# name resolve order = bcast host
domain logons = yes
domain master = yes
printing = cups
printcap name = cups
# logon path = \\%N\profiles\%u
# logon script = logon.bat
map acl inherit = yes
nt acl support = yes
passdb backend = ldapsam:ldap://127.0.0.1/
ldap admin dn = cn=admin,dc=ph-uni,dc=de
ldap suffix = dc=ph-uni,dc=de
ldap group suffix = ou=Groups
ldap user suffix = ou=Users
ldap machine suffix = ou=Computers
enable privileges = yes
add machine script = /usr/lib/mmc/add_machine_script '%u'
socket options = TCP_NODELAY SO_KEEPALIVE SO_RCVBUF=8192 SO_SNDBUF=8192
```

Remember to always check the validity of the Samba configuration with the `testparm` programm, it should not indicate any error.

Then the Samba admin password is set (for obvious reasons to the LDAP admin password):

```
root@auth:~$ smbpasswd -w <the_ldap_admin_password>
Setting stored password for "cn=admin,dc=ph-uni,dc=de" in secrets.tdb
```

Start the samba server:

```
root@auth:~$ /etc/init.d/samba start
```

Now we can install the Samba component of MMC:

```
root@auth:~$ aptitude install python-mmc-samba mmc-web-samba
```

The next step is to add the MMC schema for Samba to the OpenLDAP configuration:

```
root@auth:~$ zcat /usr/share/doc/python-mmc-base/contrib/ldap/samba.schema.gz >
/etc/ldap/schema/samba.schema
root@auth:~$ nano /etc/ldap/slapd.conf
```

Add the “include” and “index” lines, and replace the “access to attr” line, like shown below:

```
[..]
include      /etc/ldap/schema/samba.schema          # include samba schema
[..]
# Indexing options for database #1
index        objectClass      eq
index        uid,uidNumber,gidNumber,memberUid eq
index        cn,mail,surname,givenname      eq,subinitial
index        sambaSID                  eq
index        sambaPrimaryGroupSID    eq
index        sambaDomainName       eq
[..]
# allow windows users to change their passwords
access to attrs=userPassword,shadowLastChange,sambaNTPassword,sambaLMPassword
[..]
```

The LDAP daemon is thus configured to index some attributes, so it can search faster for them in its database. To let the changes take effect on the existing database, we have to reindex it. *The daemon must not run*, while we do that! All future additions to the database will be indexed when added automatically. The output to be expected is noted below:

```
root@auth:~$ /etc/init.d/slapd stop
root@auth:~$ slapindex -v
WARNING!
Running as root!
There's a fair chance slapd will fail to start.
Check file permissions!

indexing id=00000001
indexing id=00000002
```

```
indexing id=00000003
indexing id=00000004
indexing id=00000005
indexing id=00000006
indexing id=00000007          # many line like that
```

As the files were modified by the root user, we have to reset the permissions so the LDAP daemon can write to it again:

```
root@auth:~$ chown openldap:openldap /var/lib/ldap/*
root@auth:~$ /etc/init.d/slapd start
```

Every MS Windows domain has a unique identifier: its “SID”. All users, groups and machine accounts in a domain also have SIDs, which are the concatenation of the domain SID and a number (the “RID”). A valid Windows domain must have some default users and groups. You must initialize the LDAP directory with them using the smbldap-tools. But to do that you need the domain SID of the Primary Domain Controller (PDC) – usually a MS Windows, Samba server or NAS<sup>7</sup>. This means that you need the file server running now. If you have not done it already, set it up (minimalistic) now, and continue here after that.

You can get that SID by logging into the file server.

```
fileserver:~$ net getlocalsid ITCP
SID for domain ITCP is: S-1-5-21-7852459028-5424524578-4353458973
```

When logged into a client computer with access to the file server the same “SID” should be returned when querying for it:

```
client:~# net rpc getsid ITCP -S fileserver
Storing SID S-1-5-21-7852459028-5424524578-4353458973 for Domain ITCP in
secrets.tdb
```

Back on the auth-server we install a couple of management tools and get the configuration straight:

```
root@auth:~$ aptitude install smbldap-tools
root@auth:~$ zcat /usr/share/doc/smbldap-tools/examples/smbldap.conf.gz > /etc/
smbldap-tools/smbldap.conf
```

---

<sup>7</sup>see chapter 24.1.4

```
root@auth:~$ cp /usr/share/doc/smbldap-tools/examples/smbldap_bind.conf /etc/
      smbldap-tools/smbldap_bind.conf
root@auth:~$ chmod 0644 /etc/smbldap-tools/smbldap.conf
root@auth:~$ chmod 0600 /etc/smbldap-tools/smbldap_bind.conf
root@auth:~$ nano /etc/smbldap-tools/smbldap.conf
```

```
[..]
SID="S-1-5-21-7852459028-5424524578-4353458973"          # SID from PDC
sambaDomain="ITCP"                                         # domain name
ldapTLS="0"                                                 # no need for encryption on localhost
suffix="dc=ph-uni,dc=de"                                     # LDAP root node
sambaUnixIdPooldn="sambaDomainName=${sambaDomain},${suffix}"
#defaultMaxPasswordAge="45"                                    # deactivate password aging
userSmbHome=""                                              # we'll use the 'logon home'
               configuration from the file servers smb.conf
userProfile=""                                              # we'll use the 'logon path'
               configuration from the file servers smb.conf
mailDomain="mail.ph-uni.de"                                   # your mail domain
[..]
```

```
root@auth:~$ nano /etc/smbldap-tools/smbldap_bind.conf
```

```
slaveDN="cn=admin,dc=ph-uni,dc=de"   # the LDAP admin account
slavePw="secret"                      # the password of the LDAP admin account
masterDN="cn=admin,dc=ph-uni,dc=de"    # the LDAP admin account
masterPw="secret"                      # the password of the LDAP admin account
```

Now we can populate the LDAP with the necessary units for a MS Windows domain, where we get output like the following:

```
root@auth:~$ smbldap-populate -m 512 -a administrator
Populating LDAP directory for domain ITCP (S
-1-5-21-7852459028-5424524578-4353458973)
(using builtin directory structure)

entry dc=ph-uni,dc=de already exist.
entry ou=Users,dc=ph-uni,dc=de already exist.
entry ou=Groups,dc=ph-uni,dc=de already exist.
entry ou=Computers,dc=ph-uni,dc=de already exist.
adding new entry: ou=Idmap,dc=ph-uni,dc=de
adding new entry: uid=administrator,ou=Users,dc=ph-uni,dc=de
adding new entry: uid=nobody,ou=Users,dc=ph-uni,dc=de
adding new entry: cn=Domain Admins,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Domain Users,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Domain Guests,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Domain Computers,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Administrators,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Account Operators,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Print Operators,ou=Groups,dc=ph-uni,dc=de
```

```

adding new entry: cn=Backup Operators,ou=Groups,dc=ph-uni,dc=de
adding new entry: cn=Replicators,ou=Groups,dc=ph-uni,dc=de
adding new entry: sambaDomainName=ITCP,dc=ph-uni,dc=de

Please provide a password for the domain root:
Changing UNIX and samba passwords for root
New password:                               # Attention! This sets your root password!
Retype new password:                         # Attention! This sets your root password!

```

If you look into the LDAP now, you can see that more nodes have been created:

```

root@auth:~$ ldapsearch -x -W -h localhost -D cn=admin,dc=ph-uni,dc=de -b dc=ph
-uni,dc=de

```

It is very likely necessary to add the domain SID to Samba again (it needs to be stored in a local database). Check for the SID, by running and set it if necessary:

```

root@auth:~$ net getlocalsid ITCP
[2008/03/28 06:37:41, 0] utils/net.c:net_getlocalsid(583)
  Can't fetch domain SID for name: ITCP
root@auth:~$ net setdomainsid S-1-5-21-7852459028-5424524578-4353458973
root@auth:~$ net getlocalsid ITCP  # don't forget to supply your domain here,
  or it will show you only the SID of the host
SID for domain ITCP is: S-1-5-21-7852459028-5424524578-4353458973

```

Make sure that in */etc/mmc/plugins/samba.ini* the baseComputersDN is set correctly:

```

[...]
baseComputersDN = ou=Computers,dc=ph-uni,dc=de
[...]

```

The MMC is now able to use its new Samba component.

```

root@auth:~$ /etc/init.d/mmc-agent restart
Restarting Mandriva Management Console XML-RPC Agent ...
mmc-agent starting...
Plugin base loaded, API version: 5:0:1 build(572)      # MMC base plugin
Plugin samba loaded, API version: 5:0:4 build(577)      # MMC samba plugin
done.

```

You can find a “SAMBA service” section in the web interface now. Please do not edit any options in “Shares → General Options”, as this will rewrite our */etc/samba/smb.conf*, and we don’t want that!

## **28.5. LDAP-authentication and authorization (NSS-LDAP and PAM-LDAP) 261**

To set the maximum password age, run (we don't want password to expire):

```
root@auth:~$ pdbedit -P "maximum password age" -C 0
```

We need to grant the “Domain Admins” users the right to join machines to the domain:

```
root@auth:~$ net -U administrator rpc rights grant 'ITCP\Domain Admins' SeMachineAccountPrivilege  
Password:  
Successfully granted rights.
```

As we assume the CIFS-service is now performed by a NAS or file server, deactivate samba:

```
root@auth:~$ /etc/init.d/samba stop  
root@auth:~$ update-rc.d -f samba remove
```

## **28.5 LDAP-authentication and authorization (NSS-LDAP and PAM-LDAP)**

We want to be able to see the user's names locally, so we must add LDAP support to the Linux authentication system. We will install the authorization part (PAM), but not configure it properly, because there is no need for the users to log on to the server. Only on the clients it will be necessary to do that.

```
root@auth:~$ aptitude install libnss-ldap libpam-ldap  
LDAP server Uniform Resource Identifier: ldap://127.0.0.1:389/ # now  
configuring libnss-ldap  
Distinguished name of the search base: dc=ph-uni,dc=de  
LDAP version to use: 3  
LDAP account for root: cn=admin,dc=ph-uni,dc=de  
LDAP root account password: very_secret # the LDAP admin password  
Make local root Database admin: Yes # now configuring libpam-ldap  
Does the LDAP database require login?: No  
LDAP account for root: cn=admin,dc=ph-uni,dc=de  
LDAP root account password: very_secret # the LDAP admin password
```

HINT: To generate strong passwords you can use the tool "automatic password generator":

```
root@auth:~$ aptitude install apg ; apg -m 30 -M NCL -a 1
```

The NSS still needs some configuration. Find the following lines and change them accordingly:

```
root@auth:~$ nano /etc/libnss-ldap.conf
```

```
[..]
base dc=ph-uni,dc=de
uri ldap://127.0.0.1:389/
ldap_version 3
rootbinddn cn=admin,dc=ph-uni,dc=de
scope one
timelimit 5
bind_timelimit 5
bind_policy soft
nss_base_passwd          ou=Users,dc=ph-uni,dc=de?one
nss_base_shadow           ou=Users,dc=ph-uni,dc=de?one
nss_base_group            ou=Groups,dc=ph-uni,dc=de?one
nss_map_attribute uniqueMember member
[..]
```

Finally we need to activate the Name Service Switch (NSS):

```
root@auth:~$ nano /etc/nsswitch.conf
```

```
passwd:          files  ldap
group:          files  ldap
shadow:         files  ldap

hosts:          files  dns
networks:       files

protocols:      db    files
services:       db    files
ethers:         db    files
rpc:            db    files
netgroup:       ldap  [NOTFOUND=return] files nis
```

Check whether it worked, by letting the system print out all users and groups it knows. You should get all the local users/groups from */etc/passwd* respective */etc/group* and then the users/groups stored in the LDAP (as configured in */etc/nsswitch.conf*):

```
root@auth:~$ getent passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
[...]
administrator:x:0:512: administrator:/home/administrator:/bin/bash
nobody:x:999:514:nobody:/dev/null:/bin/false
```

```
root@auth:~$ getent group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
[...]
Domain Admins:*:512:administrator
Domain Users:*:513:
Domain Guests:*:514:
Domain Computers:*:515:
Administrators:*:544:
Account Operators:*:548:
Print Operators:*:550:
Backup Operators:*:551:
Replicators:*:552:
```

From now on, if you reboot the server, you will see error messages like this:

```
Starting the hotplug events dispatcher: udevdudevd[849]: nss_ldap:
could not connect to any LDAP server as cn=admin,dc=ph-uni,dc=de - Can't
    contact LDAP server
udevd[849]: nss_ldap: failed to bind to LDAP server ldap://127.0.0.1:389/: Can'
    t contact LDAP server
udevd[849]: nss_ldap: could not search LDAP server - Server is unavailable
```

This happens, because `udevd` tries to read from LDAP before it has been started. It just fails and goes on and finds its users in `/etc/passwd`. No problem here at all.

The auth server can be used now to authenticate UNIX and MS Windows users. You can continue your configuration of the Samba server now.

If everything went right, you can test your configuration by adding a user (either UNIX or Samba), and running:

```
root@auth:~$ smbclient -L localhost -U testuser1
Password:
Domain=[ITCP] OS=[Unix] Server=[Samba 3.0.24]

Sharename      Type      Comment
-----        ---       -----
IPC$          IPC       IPC Service (Samba 3.0.24) Domain=[ITCP] OS=[Unix]
                      Server=[Samba 3.0.24]

Server          Comment
-----          -----
AUTH           Samba 3.0.24
Workgroup      Master
-----          -----
ITCP           AUTH
```

Getting output similar to the above you have evidence that

- you could authenticate as a user
- no shares are open on the auth server
- the auth server is the PDC of the domain ITCP

If you install Samba on another server (we call it fileserver), or if you configure a NAS, then you can tell it to use the same configuration you used for the local Samba, except that it has to look for the LDAP server at `ldaps://auth:636/`

If you want to check that this works, run the following from the fileserver and make sure that you use the correct hostname of your auth server:

```
fileserver:~$ ldapsearch -x -W -H ldaps://auth:636/ -D cn=admin,dc=ph-uni,dc=de
-b dc=ph-uni,dc=de
```

You should see the exact same output as you get directly on the auth server.

## 28.6 Mail Users

With the MMC mail component each user can have a mail account. To make this happen we have to install the respective MMC plugins:

```
root@auth:~$ aptitude install mmc-web-mail python-mmc-mail
```

As we did several times before we add the MMC schema for mail to the OpenLDAP configuration:

```
root@auth:~$ cp /usr/share/doc/python-mmc-base/contrib/ldap/mail.schema /etc/
          ldap/schema
root@auth:~$ nano /etc/ldap/slapd.conf
```

```
[..]
include      /etc/ldap/schema/mail.schema
[..]
```

We also have to configure the plugin by editing the file `/etc/mmc/plugins/mail.ini`:

```
[main]
vDomainSupport = 1
vDomainDN = ou=mailDomains, dc=ph-uni,dc=de
destpath = /var/mmc/archives
```

Having done this, the LDAP daemon and the MMC-agent have to be restarted. The output while starting we already know, but the loading of the newly installed plugins will appear as well:

```
root@auth:~$ /etc/init.d/slapd restart
root@auth:~$ /etc/init.d/mmc-agent restart
Restarting Mandriva Management Console XML-RPC Agent: mmc-agent starting...
Registering authenticator baseldap / base.BaseLdapAuthenticator
Registering authenticator externalldap / mmc.plugins.base.externalldap.
    ExternalLdapAuthenticator
Registering provisioner externalldap / mmc.plugins.base.externalldap.
    ExternalLdapProvisioner
Registering computer manager baseldap / base.Computers
Plugin base loaded, API version: 5:0:1 build(572)
Plugin mail loaded, API version: 6:0:4 build(568)
Plugin samba loaded, API version: 5:0:4 build(577)
Selecting authenticator baseldap / base.BaseLdapAuthenticator
Authenticator baseldap successfully validated
Selecting provisioners: None
Selecting computer manager: none
done.
```

You can find a “Mail service” section in the web interface now.

## 28.7 DNS and DHCP Server

Having a local DNS server has various advantages: using WINS as naming service for MS Windows clients works, but is considered deprecated, DNS works on all platforms and applications and DNS requests can be cached locally which reduces latency.

A DHCP server is a must-have in an environment with a large number of computers. But if IPs are assigned dynamically, then it is not guaranteed to reach the computers always at the same IPs. There are two solutions for this problem: The DHCP server can update the DNS servers database after assigning an IP to a computer, or it can be configured on the DHCP server, that a certain MAC address always receive the same IP address, and that IP is saved in the DNS servers database. In both cases we can use DHCP to assign IPs to computers, and still use DNS to contact them. MMC uses the first method for achieving this goal.

MMC has a management plugin for managing a DNS and a DHCP server which we are going to install.

There are different versions of some packages (especially for bind) in the Debian and in the

Mandriva repository. In these cases we want to prefer those from the Mandriva repository, which we give higher priority by adding these lines to */etc/apt/preferences*:

```
Package: *
Pin: origin mds.mandriva.org
Pin-Priority: 1001
```

```
root@auth:~$ aptitude update
root@auth:~$ aptitude install dhcp3-server dhcp3-server-ldap bind9 mmc-web-
    network python-mmc-network
```

At the end of the installation process the system tries to start the DNS-server and the DHCP-server – the latter will fail, as further configuration is needed.

First tell the DNS-server to read its LDAP-configuration, by adding 1 line at the end of its configuration:

```
root@auth:~$ nano /etc/bind/named.conf
```

```
[..]
include "/etc/bind/named.conf.local";                      # this might be named \code
{named.conf.itcp.ph-uni.de}
include "/etc/bind/named.conf.ldap";                      # add this line
```

The file */etc/bind/named.conf.ldap* can either be empty (it just tells the DNS server to use LDAP to store domain names) or can just have the line:

```
include "/etc/bind/named.ldap/local";                      # this might be named \code
{/etc/bind/named.ldap/itcp.ph-uni.de}
```

Now make the system use the DNS server itself:

```
root@auth:~$ nano /etc/resolv.conf
nameserver 127.0.0.1
```

The DHCP server is configured as follows in order to use LDAP for storing DHCP leases:

```
root@auth:~$ nano /etc/dhcp3/dhcpd.conf
```

```
ldap-server "localhost";
ldap-port 389;
ldap-username "cn=admin,dc=ph-uni,dc=de";
ldap-password "your_password_here";           # your LDAP-admin password here
ldap-base-dn "dc=ph-uni,dc=de";
ldap-method dynamic;
ldap-debug-file "/var/log/dhcp-ldap-startup.log";
```

Yet again we have to introduce two new schemas to our LDAP server:

```
root@auth:~$ gzip -dc /usr/share/doc/python-mmcc-base/contrib/ldap/dhcp.schema.
      gz > /etc/ldap/schema/dhcp.schema
root@auth:~$ gzip -dc /usr/share/doc/python-mmcc-base/contrib/ldap/dnszone.
      schema.gz > /etc/ldap/schema/dnszone.schema
root@auth:~$ nano /etc/ldap/slapd.conf
```

```
[..]
include      /etc/ldap/schema/dhcp.schema
include      /etc/ldap/schema/dnszone.schema
[..]
index      zoneName,relativeDomainName          eq
index      dhcpHWAddress,dhcpClassData          eq
[..]
```

Now we can install the MMC component to manage DHCP and DNS.

```
root@auth:~$ /etc/init.d/slapd restart
root@auth:~$ aptitude install python-mmcc-network

root@auth:~$ nano /etc/mmcc/plugins/network.ini
```

```
[..]
[dhcp]
dn = ou=DHCP,dc=ph-uni,dc=de
[..]
[dns]
dn = ou=DNS,dc=ph-uni,dc=de
[..]
```

To make sure that when restarting the server, the LDAP daemon is started before the DNS daemon, we change the startup order:

```
root@auth:~$ update-rc.d -f slapd remove && update-rc.d slapd start 14 2 3 4 5
      . stop 86 0 1 6 .
```

```
Removing any system startup links for /etc/init.d/slapd ...
/etc/rc0.d/K80slapd
/etc/rc1.d/K80slapd
/etc/rc2.d/S19slapd
/etc/rc3.d/S19slapd
/etc/rc4.d/S19slapd
/etc/rc5.d/S19slapd
/etc/rc6.d/K80slapd
Adding system startup for /etc/init.d/slapd ...
/etc/rc0.d/K86slapd -> ../init.d/slapd
/etc/rc1.d/K86slapd -> ../init.d/slapd
/etc/rc6.d/K86slapd -> ../init.d/slapd
/etc/rc2.d/S14slapd -> ../init.d/slapd
/etc/rc3.d/S14slapd -> ../init.d/slapd
/etc/rc4.d/S14slapd -> ../init.d/slapd
/etc/rc5.d/S14slapd -> ../init.d/slapd
```

Now we can restart the mmc-agent, hopefully seeing a clear startup with all the plugins being loaded:

```
root@auth:~$ /etc/init.d/mmc-agent restart
Restarting Mandriva Management Console XML-RPC Agent: mmc-agent starting...
The default user group Domain Users does not exist. Please create it before
adding new users.
Registering authenticator baseldap / base.BaseLdapAuthenticator
Registering authenticator externalldap / mmc.plugins.base.externalldap.
    ExternalLdapAuthenticator
Registering provisioner externalldap / mmc.plugins.base.externalldap.
    ExternalLdapProvisioner
Plugin base loaded, API version: 6:0:2 build(620)      # MMC base plugin
Plugin mail loaded, API version: 6:0:4 build(568)      # MMC mail plugin
Plugin samba loaded, API version: 5:0:4 build(577)      # MMC samba plugin
Created OU ou=DHCP,dc=ph-uni,dc=de
Created OU ou=DNS,dc=ph-uni,dc=de
Created DHCP config object
The server 'auth' has been set as the primary DHCP server
Plugin network loaded, API version: 1:1:0 build(620)      # MMC network plugin
Selecting authenticator baseldap / base.BaseLdapAuthenticator
Authenticator baseldap successfully validated
Selecting provisioners: None
Selecting computer manager: none
done.
```

The DNS server works now and has to be restarted:

```
root@auth:~$ /etc/init.d/bind9 restart
Stopping domain name service...: bind.
Starting domain name service...: bind.
```

But the DHCP server won't start since we did not configure any interface.

To configure it we use the web frontend. After we have logged in, we can see the network plugins configuration panel. Use it to “Add a DNS zone”. Enter the following configuration (adjust it to your needs):

- DNS zone FQDN: itcp.ph-uni.de
- Description: ITCP PC pool
- Name server host name: auth
- Name server IP: 192.168.200.253 # your auth servers IP
- Network address: 192.168.200.0 # your networks address
- Network mask: 24
- Also manage a reverse DNS zone: yes
- Also create a related DHCP subnet: yes

After creation you are presented with an overview of all DNS zones you manage. There is only one at the moment. The auth server’s IP/hostname should be already configured in that domain. Add the auth server by entering:

- Host name: endian
- Network address: 192.168.200.254 # your firewalls IP

To get familiar with MMC you should also refer to the documentation, which can be found here:<http://mds.mandriva.org/wiki>.

Now go to the “DHCP subnets”. You’ll see, that there is already one configured for “192.168.200.0”. Let’s configure it more properly. Edit its configuration by clicking on the “Edit” icon:

- DHCP subnet address: 192.168.200.0
- Netmask: 24
- Description: ITCP PC pool
- Authoritative: yes
- Broadcast address: 192.168.200.255
- Domain name: itcp.ph-uni.de
- Routers: 192.168.200.1
- Domain name servers: 192.168.200.250
- NTP servers:
- WINS servers:

- WINS resolution and registration method:

You can leave the fields of the “Other DHCP options” empty, but you must enter the lease times:

- Minimum lease time: 60
- Default lease time: 3600
- Maximum lease time: 86400
- Dynamic pool for non-registered DHCP clients: yes
- IP range start: 192.168.200.1
- IP range end: 192.168.200.200

Now you need to restart the DHCP server, for the changes to take place. Use the “Network services management” link on the left, to come to a screen where you can (clicking on the icons on the right) start, stop, restart, reload the DNS and DHCP server, and read their logs.

If everything worked, you’ll find entries in */var/log/syslog* from *named* and *bind*. If you look at the output of *lsof* you should find something similar to:

```
[..]
named    6629      bind    20u   IPv6  16956      TCP  *:53  (LISTEN)
named    6629      bind    21u   IPv4  16961      TCP  127.0.0.1:53  (LISTEN)
named    6629      bind    22u   IPv4  16963      TCP  192.168.200.253:53  (LISTEN
)
named    6629      bind    23u   IPv4  16964      TCP  127.0.0.1:953  (LISTEN)
named    6629      bind    24u   IPv6  16965      TCP  [::]:953  (LISTEN)
named    6629      bind    512u  IPv6  16955      UDP  *:53
named    6629      bind    513u  IPv4  16960      UDP  127.0.0.1:53
named    6629      bind    514u  IPv4  16962      UDP  192.168.200.253:53
dhcpd3  6644      root     5u   IPv4  17098      UDP  *:67
dhcpd3  6644      root     6u   IPv4  17088      TCP
127.0.0.1:38884->127.0.0.1:389  (ESTABLISHED)
```

## 28.8 Linux Clients

In this chapter the basic steps needed to let a linux computer authenticate and authorize the users against the LDAP auth server and mount the file server’s shares as well as the users’ home directories using *pam\_mount*.

### 28.8.1 Configure LDAP connections

We want to use a Name Service Switch (NSS) that allows the system to use configuration provided by different sources, in our case local files (*/etc/passwd*, */etc/group*) and the LDAP, for

getting user/group information. Additionally we want PAM (Pluggable Authentication Modules) to use the LDAP too. Thus we have to install the following and configure them correctly:

```
root@workstation:~$ aptitude install libnss-ldap libpam-ldap
root@workstation:~$ vim /etc/ldap.conf
```

```
base dc=ph-uni,dc=de
uri ldaps://192.168.0.253:636
ldap_version 3
scope one
timelimit 5
bind_timelimit 5
bind_policy soft
nss_base_passwd ou=Users,dc=ph-uni,dc=de?one
nss_base_shadow ou=Users,dc=ph-uni,dc=de?one
nss_base_group ou=Groups,dc=ph-uni,dc=de?one
nss_map_attribute uniqueMember member
```

```
root@workstation:~$ vim /etc/ldap/ldap.conf
```

```
BASE      dc=ph-uni,dc=de
URI       ldaps://192.168.0.253:636/
TLS_REQCERT never
```

```
root@workstation:~$ vim /etc/nsswitch.conf
```

```
passwd:          files ldap
group:           files ldap
shadow:          files ldap

hosts:            files dns
networks:        files

protocols:       db files
services:         db files
ethers:          db files
rpc:              db files
netgroup:        ldap [NOTFOUND=return] files nis
```

Now after restarting the libnss-ldap daemon we can check whether it works properly:

```
root@workstation:~$ /etc/init.d/libnss-ldap restart
root@workstation:~$ getent passwd
root@workstation:~$ getent shadow
root@workstation:~$ getent group
```

We should get not only the local users and groups but the ones stored in the LDAP additionally. We have seen this before in section 28.5.

### 28.8.2 Configure PAM

We are going to configure the pam.d now to use the LDAP and insert the following lines to the files:

```
root@workstation:~$ cd /etc/pam.d  
root@workstation:/etc/pam.d$ vim common-account
```

```
[...]  
account sufficient pam_ldap.so  
account required pam_unix.so  
[...]
```

```
root@workstation:/etc/pam.d$ vim common-auth
```

```
[...]  
auth sufficient pam_ldap.so  
auth required pam_unix.so nullok_secure try_first_pass  
[...]
```

```
root@workstation:/etc/pam.d$ vim common-password
```

```
[...]  
password sufficient pam_ldap.so  
password required pam_unix.so nullok obscure min=4 max=8 md5  
[...]
```

```
root@workstation:/etc/pam.d$ vim common-session
```

```
[...]  
# session optional pam_foreground.so  
session sufficient pam_ldap.so  
[...]
```

```
root@workstation:/etc/pam.d$ vim login
```

```
[...]
auth optional pam_mount.so
auth sufficient pam_unix.so nullok_secure try_first_pass
auth required pam_ldap.so try_first_pass

session optional pam_mount.so
auth optional pam_group.so
[...]
```

```
root@workstation:/etc/pam.d$ vim gdm
```

```
[...]
auth optional pam_mount.so
auth sufficient pam_unix.so nullok_secure try_first_pass
auth required pam_ldap.so try_first_pass

session optional pam_mount.so
auth optional pam_gnome_keyring.so

auth optional pam_mount.so
[...]
```

### 28.8.3 Setup pam\_mount

The `pam_mount` module is designed for environments with central file servers that a user wishes to mount on login and unmount on logout, where many users can logon and where statically mounting the entire `/home` from a server is a security risk, and listing all possible volumes in `/etc/fstab` is not feasible.

The configuration file for `pam_mount` is located in `/etc/security/`.

```
root@workstation:~$ vim /etc/security/pam_mount.conf.xml
```

```
[...]
<!-- Volume definitions -->
<volume user="*" fstype="cifs" server="192.168.200.220" path="%USER"
        mountpoint="~" />
```

The public shares can be mounted then for example like this:

```
mount -t cifs -o user=guest,guest //dss/WinXP /mnt
or
mount -t smbfs -o user=guest,sec=none //dss/WinXP /mnt
```

## 28.9 Windows Clients

With Windows XP Professional it is quite simple to join a domain. Windows XP Home Edition is not designed to join domains. Just follow these steps. You will need the domain administrator's username and password:

1. Click "Start", right-click "My Computer", and then click "Properties".
2. Click the "Computer Name" tab, and then click "Change".
3. Type the new domain in the Domain dialog box.
4. If the "Computer Account" does not yet exist on the Domain server, then you will be prompted for a Username and password. Supply the domain administrator's credentials here.
5. Click OK, and then restart the computer.

The same process applies to Windows Vista clients as well.

# Proxy Server

---

## Contents

<b>29.1 Motivation</b>	.....	275
<b>29.2 Definition</b>	.....	275
<b>29.3 Forward proxy / Reverse Proxy</b>	.....	276
<b>29.4 Non-transparent / Transparent Proxy</b>	.....	276
<b>29.5 Examples</b>	.....	276
29.5.1 Caching debian packages	.....	276
29.5.2 Transparent Web proxy	.....	278

---

This chapter explains what a proxy server is and what it can be used for. The example section focuses on the usage of proxy servers as web cache.

## 29.1 Motivation

The typical network architecture of an IT-center as shown in figure 29.1 connects many clients through a high speed LAN to a router which acts as the gateway to the Internet. The link between the gateway and the ISP is usually much slower than the connections inside the LAN. Also, sometimes this up-link is charged according to data volume. These reasons make it desirable to keep the data volume on the up-link as low as possible. To accomplish this, a concept well known in computer science can be used: caching. A proxy server can act as a cache between the private network and the Internet.

## 29.2 Definition

In computer networks, a proxy server is a server (a computer system or an application program) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server evaluates the request according to its filtering rules. For example, it may filter traffic by IP address or protocol. If the request is validated by the filter, the proxy provides the resource by connecting to the relevant server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it 'caches' responses from the remote server, and returns subsequent requests for the same content directly.[[wik 2010b](#)]

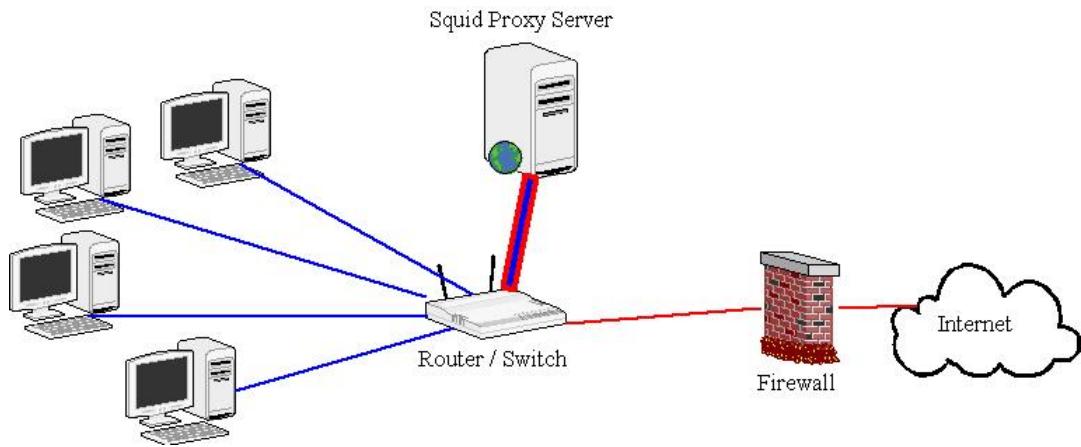


Figure 29.1: Squid acting as a web proxy.

### 29.3 Forward proxy / Reverse Proxy

A forward proxy is usually a proxy which is located in a private network and provides access to the Internet for the machines on this network. It can be used for many purposes including filtering, logging, scanning and caching the traffic passing it. A reverse proxy is facing the Internet and acts as a front-end to one or many servers on a private network. It can be used for load-balancing, authentication, decryption or caching.

### 29.4 Non-transparent / Transparent Proxy

In a heterogeneous environment with many clients, configuring each client to use a proxy server can be infeasible. To solve this problem a transparent proxy server can be used. A transparent proxy is invisible (transparent) to the clients. It is also sometimes referred to as an intercepting proxy, because it intercepts the clients requests without their knowledge. Its configuration is completely done on the server and requires no configuration on the clients. In contrast, a non-transparent proxy is a proxy server which needs explicit configuration on the client side.

### 29.5 Examples

This section focuses on concrete examples on how to configure a proxy server in different situations. In the first example, a caching server for debian package files will be configured. The second example describes the steps needed to be taken to install a transparent proxy acting as a web cache for a LAN.

#### 29.5.1 Caching debian packages

In a computer center, it is common to update or install new software on all clients on a regular basis. In the case of debian based clients, this means that each client needs to download the

same set of debian package files. Without any special configuration, updating each client will cause it to download all the package files from the configured debian mirror on the Internet. This results in an enormous demand in bandwidth which can be decreased considerably by caching the requested packages. Many solutions exist which fulfill this task, two of them will be presented here: *squid-deb-proxy*<sup>1</sup> and *apt-cacher-ng*<sup>2</sup>.

### Squid-deb-proxy

*Squid-deb-proxy* is based on *Squid*<sup>3</sup>, a general web caching proxy. It is only available for Ubuntu and is mostly a preconfigured squid optimized for caching debian packages. Its configuration file is located in `/etc/squid-deb-proxy/squid-deb-proxy.conf` and log files are stored in `/var/log/squid-deb-proxy/`. The default configuration allows access to the proxy from all private networks and localhost. The proxy will only allow access to certain ubuntu mirrors which can be configured in the file `/etc/squid-deb-proxy/mirror-dstdomain.acl`. Package files will reside in the cache for a maximum of 90 days if the associated http response does not contain a expire time header entry. This behaviour is configured by a `refresh_pattern`<sup>4</sup> directive for debian package files. Additionally to the package `squid-deb-proxy`, which contains the Squid server and its configuration, the package `squid-deb-proxy-client` can be installed on the clients. This package leverages *Avahi*<sup>5</sup> to automatically configure *Apt* programs to use any existing proxy server in the local network. If the `squid-deb-proxy-client` is not installed, the clients must be configured manually as described in section 29.5.1.

### Apt-cacher-ng

*Apt-cacher-ng* is a special purpose caching proxy optimized for caching debian packages. It is available for Debian and Ubuntu. Its main advantage over *squid-deb-proxy* is its ability to rewrite requests for the same debian package from different mirrors. If, for example one client requests the package

`http://de.archive.ubuntu.com/ubuntu/pool/main/b/binutils/binutils_2.20.deb` and another client request the package

`http://en.archive.ubuntu.com/ubuntu/pool/main/b/binutils/binutils_2.20.deb`, *squid-deb-proxy* would create two cache entries, even though both requests may refer to the same file `binutils_2.20.deb`. The reason for this is that the requests differ in the host name and *squid-deb-proxy* can only match requests with the exact same URL. Using *apt-cacher-ng* a list of mirrors can be specified where requests to these mirrors will be rewritten to refer to only one common mirror. This way, the rate of cache hits, especially in environments where clients contain different mirror configurations, can be greatly increased.

*Apt-cacher-ng* is configured in the file `/etc/apt-cacher-ng/acng.conf` and logs its activity to `/var/log/apt-cacher-ng/apt-cacher.log`. The list of mirrors to which requests should be rewritten is located in `/etc/apt-cacher-ng/ubuntu_mirrors` and `/etc/apt-cacher-ng/debian_mirrors`.

The mirror that will be used to serve rewritten requests is specified in the file

---

<sup>1</sup><http://www.linuxjournal.com/content/presenting-squid-deb-proxy-speed-your-update-downloads>

<sup>2</sup><http://freshmeat.net/projects/acng>

<sup>3</sup><http://www.squid-cache.org>

<sup>4</sup>[http://www.squid-cache.org/Doc/config/refresh\\_pattern/](http://www.squid-cache.org/Doc/config/refresh_pattern/)

<sup>5</sup><http://avahi.org>

`/etc/apt-cacher-ng/backends_ubuntu.default` and `/etc/apt-cacher-ng/backends_debian.default`. The configuration of the clients to use *apt-cacher-ng* is described in the next section.

### Configuring the client

To configure all *Apt* programs like `aptitude`, `apt-get` or `dpkg` to use a proxy server, the following file must be added to the directory `/etc/apt/apt.conf.d/`.

```
# /etc/apt/apt.conf.d/02proxy
Acquire::http { proxy "http://<ip_of_proxy_server>:3142"; }
# The following line is only for debug purposes
# Debug::Acquire::http "true";
```

The first line configures all *Apt* programs to use the proxy server at the specified IP address and port number. Of course, the port number must match the one the proxy server uses to listen for incoming connections. The second line is thought only for debugging the configuration and should be switched off in a production environment. It will print some information related to the HTTP download of packages when using *Apt* to download or install packages.

#### 29.5.2 Transparent Web proxy

This section discusses how to set up a transparent squid web proxy. Two scenarios will be used, the first where the proxy is located on the same host as the standard gateway and the second where the proxy is located on a separate host inside the private network. In both examples, `iptables` is used to rewrite all Tcp/Ip packages destined to port 80. This way, HTTP traffic can be transparently intercepted.

More information about squid configurations can be found on <http://wiki.squid-cache.org/ConfigExamples>.

##### Scenario I

This scenario assumes that the proxy server is installed on the same host as the standard gateway of the LAN the proxy should be serving. That means that all outgoing traffic already passes the host the proxy server is installed. All that has to be done is to redirect packages destined to port 80 to the port the proxy is listening on. The following file is a bash script that sets up the required `iptables` rules to achieve that behavior.

```
#!/bin/bash
SQUIDIP=<public ip of squid box>

# clear table
```

```
iptables -t nat -F
# accept traffic from squid
iptables -t nat -A PREROUTING -s $SQUIDIP -p tcp --dport 80 -j ACCEPT
# redirect port 80 to port 3127
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3127
```

It is assumed that *Squid* listens on port 3127. The second *iptables* rule redirects all packages destined to port 80 to port 3127. The first *iptables* rule ensures that traffic that already originate from the proxy will not be redirected again to avoid a forwarding loop.

For *squid* to act as a transparent proxy, the following line has to be added to the file `/etc/squid/squid.conf`<sup>6</sup>

```
http_port 3127 transparent
```

## Scenario II

In this scenario, the proxy server is located on a separate host than the standard gateway. Thus traffic through the gateway must be first routed to the proxy host. To achieve this, a two phase approach is used which incorporates *iptables* to mark packages destined to port 80 and *policy based routing* to set up a different route for the marked packages. This configuration has to be done on the gateway. The configuration on the proxy host is the same as in scenario I.

The following bash script sets up the required *iptables* rules to mark all packages destined to port 80.

```
#!/bin/bash

SQUIDIP=<public ip of squid box>

# clear table
iptables -t mangle -F

# accept traffic from squid
iptables -t mangle -A PREROUTING -s $SQUIDIP -p tcp --dport 80 -j ACCEPT
# mark every packages to port 80
iptables -t mangle -A PREROUTING -p tcp --dport 80 -j MARK --set-mark 2
```

As in scenario I, the first *iptables* rule ensures that packages originating from the proxy host will not be marked again. The value '2' in the second rule is chosen arbitrary and can be any number between 1 and 255. It must however match the number used in the second step when setting up the routing rules.

<sup>6</sup>[http://www.squid-cache.org/Doc/config/http\\_port/](http://www.squid-cache.org/Doc/config/http_port/)

Now, policy based routing is used to route the marked packages to the proxy host. At first, a new routing table must be created by adding a free number and a name to the file `/etc/iproute2/rt_tables`:

```
echo "100 proxy" >> /etc/iproute2/rt_tables
```

In this example 100 is used but it is important to use a number which is not yet occupied.

The following line will cause the kernel to use the new routing table for all marked packages.

```
ip rule add fwmark 2 table proxy
```

Finally, the next line adds the proxy host as the standard gateway to the new routing table. This causes all marked packages to be send to the proxy host.

```
ip route add default via $SQUIDIP table proxy
```

Please refer to the man page of the `ip` command for more information on policy based routing.

# Bibliography

- [tru 2009] *Acronis Website*. <http://www.acronis.eu/homecomputing/products/trueimage/>, 2009. 207
- [wik 2010a] *Wikipedia - The Free Encyclopedia - Desktop environment*. [http://en.wikipedia.org/wiki/Desktop\\_environment](http://en.wikipedia.org/wiki/Desktop_environment), 2010. 41
- [wik 2010b] *Wikipedia - The Free Encyclopedia - Proxy server*. [http://en.wikipedia.org/wiki/Proxy\\_server](http://en.wikipedia.org/wiki/Proxy_server), 2010. 275
- [wik 2010c] *Wikipedia - The Free Encyclopedia - Window manager*. [http://en.wikipedia.org/wiki/Window\\_manager](http://en.wikipedia.org/wiki/Window_manager), 2010. 41
- [wik 2010d] *Wikipedia - The Free Encyclopedia - X display manager (program type)*. [http://en.wikipedia.org/wiki/X\\_display\\_manager\\_%28program\\_type%29](http://en.wikipedia.org/wiki/X_display_manager_%28program_type%29), 2010. 40
- [wik 2010e] *Wikipedia - The Free Encyclopedia - X Window System*. [http://en.wikipedia.org/wiki/X\\_Window\\_System](http://en.wikipedia.org/wiki/X_Window_System), 2010. 39





Upper row (standing), from left to right:

Mr. George Eskander Hussein Ajam – Babylon University  
Mr. Muhanad Mohammed Kadum Al-Zwuiany – Thi Qar University  
Mr. Emad Adil Dawood – Dohuk University  
Mr. Ihab Ahmed Najm Jabor – Tikrit University  
Mr. Mohammed Baqer Mohammed Kamel – Al Qadisiya University  
Mr. Alaa Khalaf Hamoud – University of Basrah  
Mr. Bahaa Qasim Mohammed Al-Musawi – University of Kufa  
Mr. Polla Abdul-Hamid Fattah – University of Salahaddin  
Mr. Christoph Herbst – ZiiK/TU Berlin  
Mr. Xun Zhou – ZiiK/TU Berlin  
Mr. Ralph Magnus – ZiiK/TU Berlin  
Mrs. Raghdah Jameel Mahmood Al-Shaikhli – University of Technology Baghdad  
Mrs. Shene Jalil Jamal – University of Sulaimani  
Mrs. Zayneb Raed Ahmed Al-Rubaye – University of Baghdad

Lower row (sitting), from left to right:

Mr. Daniel Tippmann – ZiiK/TU Berlin  
Mr. Daniel Tröder – ZiiK/TU Berlin  
Mr. Firas Mohammed Salih Whab – University of Mosul  
Mrs. Huda Mohammed Salih Mohammed Kadim – University of Diyala