Linux for Operations

Paul Cobbaut Andy Van Maele Thomas Parmentier
Bert Van Vreckem

September 18, 2024

Contents

Feel free to contact the author(s):

- Paul Cobbaut (Netsec BVBA): paul.cobbaut@gmail.com, https://cobbaut.be/
- Bert Van Vreckem (HOGENT): http://github.com/bertvv

This copy was generated on September 18, 2024.

This book is used as the syllabus for the course "Linux" for the Bachelor of Applied Computer Science at the HOGENT, Belgium. The contents are based on the Linux Training book series by Paul Cobbaut, with updates and additions written by the HOGENT Linux team.

This book is aimed at students specialising in the Operations/System Administration track that already have some basic knowledge of Linux.

More information and free .pdf available at https://hogenttin.github.io/linux-training-hogent/.

0.1. Conventions used

The contributors to this work have taken great care that the examples with command line interactions are correct and work as expected. However, sometimes the output of a command may differ slightly from the examples in this book. This can be due to differences in the version of the software, the operating system, the environment in which the command is executed, or the specific state of the system at the time of execution.

Some Linux distributions may have commands that behave differently than their counterparts in other distributions. This is especially true for the package management commands.

Command line examples are shown in a monospaced font with a prompt that indicates the user and hostname in the form `user@hostname:current_directory$`. For example:

```
student@debian:~$ ls
root@linux:~# ls
```

We follow the following conventions:

- Regular user vs root:
  - A regular user prompt is shown as `$`, the user name is generally `student`.
  - A root prompt (with elevated privileges) is shown as `#`.
- The linux distribution is indicated by the host name:
  - If the command should work on any Linux distribution, the hostname is `linux`.
  - If the command is specific to a certain distribution, the hostname is the name of that distribution, e.g. `debian`, `ubuntu`, `rhel`, etc.
  - If you see `el` as the host name (short for Enterprise Linux), you can assume that it was tested on an Alma Linux machine and that it should work on any Red Hat-based distribution.
- Commands that are run on a prompt that is not necessarily a Linux system (e.g. you're running Linux in a VM on a Windows host) are shown with a generic `>` prompt, e.g. `> winget install Git.Git`.

0.2. Reporting errors

Did you find an error in this book, or is the output of a command considerably different from what is shown? Please report it by creating an issue on the linux-training-hogent GitHub repository.

Part I.

First Linux VM

1. distributions

(Written by Paul Cobbaut, https://github.com/paulcobbaut/ with contributions by Bert Van Vreckem https://github.com/bertvv/)

This chapter gives a short overview of current Linux distributions.

A Linux `distribution` is a collection of (usually open source) software on top of a Linux kernel. A distribution (or short, distro) can bundle server software, system management tools, documentation and many desktop applications in a `central secure software repository`. A distro aims to provide a common look and feel, secure and easy software management and often a specific operational purpose.

Let's take a look at some popular distributions.

1.1. Linux and GNU

The Linux Kernel project was started by Linus Torvalds in 1991 while he was a computer science student. He wanted to run a UNIX-like operating system on his own PC. Now, a kernel in itself is not a complete operating system. The kernel does not provide a terminal, tools to manage files, etc. However, the GNU project (which stands for *GNU's Not UNIX*), started by Richard Stallman, had been working on a complete operating system since 1983. The GNU project had a lot of the necessary tools and libraries to make a complete POSIX-compliant operating system, a.o. the GNU Compiler Collection (GCC), the GNU C Library (glibc), the GNU Core Utilities (coreutils), the GNU Bash shell, etc. They were also working on a kernel, called GNU Hurd, but development was prohibitively slow. Indeed, it was not until 2015 that the Hurd kernel was ready to be actually used.

Long story short, the Linux kernel in combination with the GNU tools and libraries made a complete operating system. This is why the operating system is often referred to as *GNU/Linux*. Both Linux as the GNU projects are open source and released under the GNU General Public License. This made it easy for third parties to redistribute GNU+Linux and add other compatible (open source) software packages to form a complete operating system with everything an end user needs to be productive on the computer. This is what we call a *Linux distribution*. The oldest still active distribution is Slackware, which was started in 1993 by Patrick Volkerding. Since then, many distributions have been created, each with their own goals and target audience. Some distributions (or distro's in short) are built from the ground up, but others are based on existing distributions, leading to large "families" of like-minded distro's.

Writing a comprehensive overview of all Linux distributions is way beyond the scope of this course, but it is useful to know about some of the main ones. If you want to know more about a specific distribution, you can check out the DistroWatch website, which is a great resource for information about Linux distributions.

1.2. Package management

One of the central and identifying components of a Linux Distribution is the default selection of software and the package management system to install, update and remove software. For most applications, there is choice in the open source world, so different distributions will

make different decisions on what to include and what to avoid. Sometimes this is regrettably the cause of dispute and drama in the Linux community, but on the other hand, it is also the driver of a lot of innovation and diversity and it empowers the user with a lot of freedom of choice and control.

The package manager was actually one of the most important innovations that Linux pioneered in. It is a system that keeps track of all the software installed on a computer and allows the user to select and install new applications from online package repositories. Hotfixes or new releases of the software included in a distribution are made available in these repositories and can be downloaded and installed with a single command. This makes it very easy to keep a Linux system up to date and secure. When Apple introduced the App Store in 2008, it was actually a latecomer to the concept of a central secure software repository.

The concept of an open source package repository also enables reuse of software and libraries. Applications don't have to write their own code to do things like read and write files, manage memory, etc. They can use libraries that are already available on the system and that are used by other applications. The package manager also takes care of dependencies, which are other software packages that are required for the software to work. This makes it very easy to install complex software with a single command.

1.3. The Red Hat family of distributions

Red Hat is one of the first commercial companies that successfully leveraged open source software as a business strategy. They started in 1993 and grew in the next decades to become a billion dollar company. In 2019, Red Hat was acquired by IBM for 34 billion dollars and it still operates as an independent subsidiary.

The flagship product of Red Hat is Red Hat Enterprise Linux, or RHEL in short. RHEL is a commercial Linux distribution, but on release, the source code is made available. The business model of Red Hat is based on selling support contracts.

RHEL is a stable and secure operating system, with long support cycles, which is why it is widely used in enterprise environments where the stability of IT infrastructure is of paramount importance. Enterprise software vendors that target Linux as a platform, usually certify their software to run on RHEL. This is why RHEL is often used in data centers, cloud environments and other mission-critical systems.

In order to innovate on the RHEL platform, Red Hat is also involved in the development of the Fedora distribution. Fedora is a community-driven project that aims to be a cutting-edge, free and open source operating system that showcases the latest in free and open source software. It is used as a testbed for new technologies that will eventually make their way into RHEL. Fedora has a release cycle of 6 months. Where RHEL is particularly suited as a server operating system, Fedora is an excellent choice as a desktop operating system for power users and IT professionals.

Since RHEL is open source, it is in principle possible to create a compatible clone of RHEL, albeit without the support and without Red Hat branding. This is exactly what the CentOS project did for years. CentOS used to be a community driven project that aimed to be 100% (*bug-for-bug*) compatible with RHEL and based on the released source code of all software included in RHEL. However, in 2014, Red Hat acquired the CentOS project, and later, they announced that CentOS Linux was going to be replaced by CentOS Stream, which is a rolling release distribution "upstream" of RHEL. This means that CentOS Stream now takes the place between Fedora and RHEL, and it is no longer a 100% compatible clone of RHEL anymore.

This incensed many users and organizations that relied on CentOS as a free and compatible alternative to RHEL. The CentOS project was forked, and the Rocky Linux project was started by Gregory Kurtzer, who was also one of the original founders of CentOS. The goal of Rocky Linux is to be a 100% compatible replacement for CentOS Linux. Likewise, AlmaLinux was started by CloudLinux, another company that was involved in the CentOS project. These RHEL-like distributions are sometimes referred to as "Enterprise Linux" or EL.

Distinctive features of the Red Hat family of distributions are:

- The use of the `RPM` package format (Red Hat Package Management) and the `dnf` package manager
- The `systemd` init system
- The `firewalld` firewall management tool
- The *SELinux* security framework
- The *Anaconda* installer
- The *Cockpit* web-based management interface
- Their own container runtimes, *runc* and *crun* and management tools *podman* and *buildah* (instead of Docker)

Oracle Enterprise Linux is Oracle's commercial Linux distribution, put in the market as a direct competitor to RHEL. Scientific Linux was a community driven project that was used by scientific institutions like CERN and Fermilab, but it was discontinued in 2021. The final maintenance window for Scientific Linux 7 is June 30, 2024. After that, users are advised to migrate to AlmaLinux. The Amazon Linux distribution is a RHEL-like distribution that is used as the default operating system for Amazon Web Services (AWS) EC2 instances.

1.4. The Debian family of distributions

There is no company behind Debian. Instead there are thousands of well organised developers that elect a *Debian Project Leader* every two years. Debian is seen as one of the most stable Linux distributions. It is also the basis of every release of the well-known Ubuntu (see below). Debian comes in three versions: stable, testing and unstable. Every Debian release is named after a character in the movie Toy Story.

Canonical, a company founded by South African entrepreneur Mark Shuttleworth, started sending out free compact discs with *Ubuntu Linux* in 2004 and quickly became popular for home users (many switching from Microsoft Windows). Canonical wants Ubuntu to be an easy to use graphical Linux desktop without need to ever see a command line. Of course they also want to make a profit by selling commercial support for Ubuntu. Ubuntu is known for their *Long Term Support* (LTS) releases, which are supported for 5 years (or 10 years for a fee). Intermediate releases come out every 6 months (in April and October) and are supported for 9 months. Releases are named after the year and month of the release, e.g. 19.10 for October 2019. LTS releases come out every even year in April, e.g. 22.04 and 24.04. Canonical also has the reputation of going their own way and doing things differently from the rest of the Linux community. For example, they developed their own init system, Upstart (which was later abandoned and replaced by systemd), and their own display server, Mir (which was later replaced by Wayland), a desktop environment (Unity, later replaced with Gnome), etc. Some of these decisions were controversial and have led to a lot of criticism, but the strength of the open source community lies precisely in the freedom to make different choices, which is a driver for innovation.

Distinctive features of the Debian family of distributions are:

- The use of the `deb` package format and the `apt` package manager (Advanced Package Tool)
- The `systemd` init system
- The `ufw` firewall management tool
- The *AppArmor* security framework
- The *Debian-installer* installer
- The Docker container runtime and management tools

Linux Mint, Edubuntu and many other distributions with a name ending on -buntu are based on Ubuntu and thus share a lot with Debian. Kali Linux is another Debian-based distribution that is specifically designed for digital forensics and penetration testing. It comes with a lot of pre-installed tools for hacking and security testing. Kali is not suitable for daily use as a

desktop operating system, but it is very popular among security professionals and hobbyists. The popular mini-computer Raspberry Pi has its own Debian-based distribution called Raspberry Pi OS.

1.5. Notable "independent" distributions

Apart from the two big families of distributions, i.e. Red Hat and Debian families, there are many other distributions that are not based on either of these. Some of the most notable ones are:

- Alpine Linux: an independent non-commercial, general purpose distribution with a focus on security and simplicity. Alpine Linux is very small and lightweight, and it is often used in containers.
- Arch Linux: another independent general purpose distribution. Arch Linux is a rolling release distribution, which means that you install it once and then continuously update individual packages when new versions become available. The distribution itself does not have an overarching (see what I did there?) release cycle. Arch has its own package manager, Pacman. One of the most notable features of Arch Linux is its outstanding documentation, which is very extensive and well written and even quite useful for users of other distributions. Installing Arch Linux is not as straightforward as installing other distributions: you start with a minimal system with the kernel and a shell, and then you build up the system to your own liking. This is not for novice users, but it is a great way to learn about the inner workings of a Linux system.
- openSUSE: a general purpose community driven distribution that is sponsored by SUSE, a German company that also offers commercial support for derivative distro's SUSE Linux Enterprise Server (SLES) and Desktop (SLED). openSUSE is known for its YaST (Yet another Setup Tool) configuration tool, which is a central place to configure many aspects of the system. openSUSE comes in two flavours: Leap and Tumbleweed. Leap is a regular release distribution with a fixed release cycle, while Tumbleweed is a rolling release distribution.

1.6. Which to choose?

If you ask 10 people what the best Linux distribution is, chances are that you will get 20 different answers. Posting it as a question on a forum may lead to a discussion that goes on for weeks or months, if not years. You will get a lot of passionate and sometimes even insightful opinions, but in the end you won't be none the wiser. So giving good advice that is universally applicable is very hard, indeed.

Below are some very personal opinions (albeit informed by experience) on some of the most popular Linux distributions. Keep in mind that any of the below Linux distributions can be a stable server and a nice graphical desktop client.

Distribution name	Reason(s) for using
AlmaLinux	You want a stable Red Hat-like server OS without commercial support contract.
Arch	You want to know how Linux *really* works and want to take your time to learn.
Debian	An excellent choice for servers, laptops, and any other device.
Fedora	You want a Red Hat-like OS on your laptop/desktop.
Kali	You want a pointy-clicky hacking interface.

Distribution name	Reason(s) for using
Linux Mint	You want a personal graphical desktop to play movies, music and games.
RHEL	You are a manager and need good commercial support.
RockyLinux	You want a stable Red Hat-like server OS without commercial support contract.
Ubuntu Desktop	Very popular, suited for beginners and based on Debian.
Ubuntu Server	(LTS particulary) You want a Debian-like OS with commercial support.

When you are new to Linux, and are looking for a distribution with a graphical desktop and all the tools that you need as a daily driver, check out the latest Linux Mint (suitable for computer novices and experienced computer users alike) or Fedora (recommended for power users and IT professionals).

If you only want to practice the Linux command line, or are interested in the use of Linux as a server, then install a VM with the latest release of either Debian stable and/or AlmaLinux (without graphical interface)[1].

As you gain experience, you can try out other distributions and see what you like best. Good luck on your journey and enjoy the ride!

[1] Remark that this advice was originally written in 2015 and basically still holds in 2024. The only amendment is that AlmaLinux has taken the place of CentOS as a recommendation for a server OS.

2. Using a prebuilt Linux VM

(Written by Bert Van Vreckem, https://github.com/bertvv)

In this chapter, you will learn how to acquire pre-built images for a Linux VM, import them into VirtualBox and start using them.

In order to start using Linux, it is not necessary to reformat or repartition your computer and install Linux as the operating system, or even in a dual-boot configuration. Instead, you can install Linux in a virtual machine (VM) on your existing computer. This is a safe and easy way to get started with Linux.

These days, it's not even necessary to go through the installation process. Although installing Linux is quite easy and can be a useful learning experience, in order to get started quickly, you can simply download a one of the many available pre-built images.

We will use VirtualBox as the virtualization platform. VirtualBox is a free and open-source desktop virtualization application maintained by Oracle. It is available for Windows, (Intel based) macOS, and Linux and provides a similar experience on all platforms. Before you start, we expect you to have VirtualBox installed on your computer. If you don't have it installed yet, you can download it from the VirtualBox website or install it using a package manager for your operating system.

2.1. Downloading a Linux VM image

Start by going to the OSBoxes website. This website is the initiative computer science student only identified under the name or handle *Umair*. The website offers a variety of ready-to-use Linux/Unix guest operating systems for VirtualBox or VMware.

Browse the website or use the search box to find a Linux distribution that you would like to try. In this example, we will use a Linux Mint image.

Scroll down to the download section, select the VirtualBox tab and click the download link for the Cinnamon Version.

The downloaded file is in the .7z (7zip) format that contains a .vdi (Virtual Disk Image) file. VirtualBox can use this file as the hard disk for a virtual machine. First, we'll need to create that virtual machine.

2.2. Creating a VirtualBox VM

Start VirtualBox and click the *New* button. In the dialog that appears, enter a name for the virtual machine, e.g. *LinuxMint*. In our case, that's the only setting that needs to be changed in this step.

Click *Next* to proceed to the next step. In the next dialog, select the amount of memory (RAM) that you want to allocate to the virtual machine. The default value is 2048 MB, which should be sufficient. However, if you have more memory available, you can increase this value. You can also assign CPU cores to the virtual machine. The default value of 1 is fine, but by increasing this value, you can improve the performance of the virtual machine.

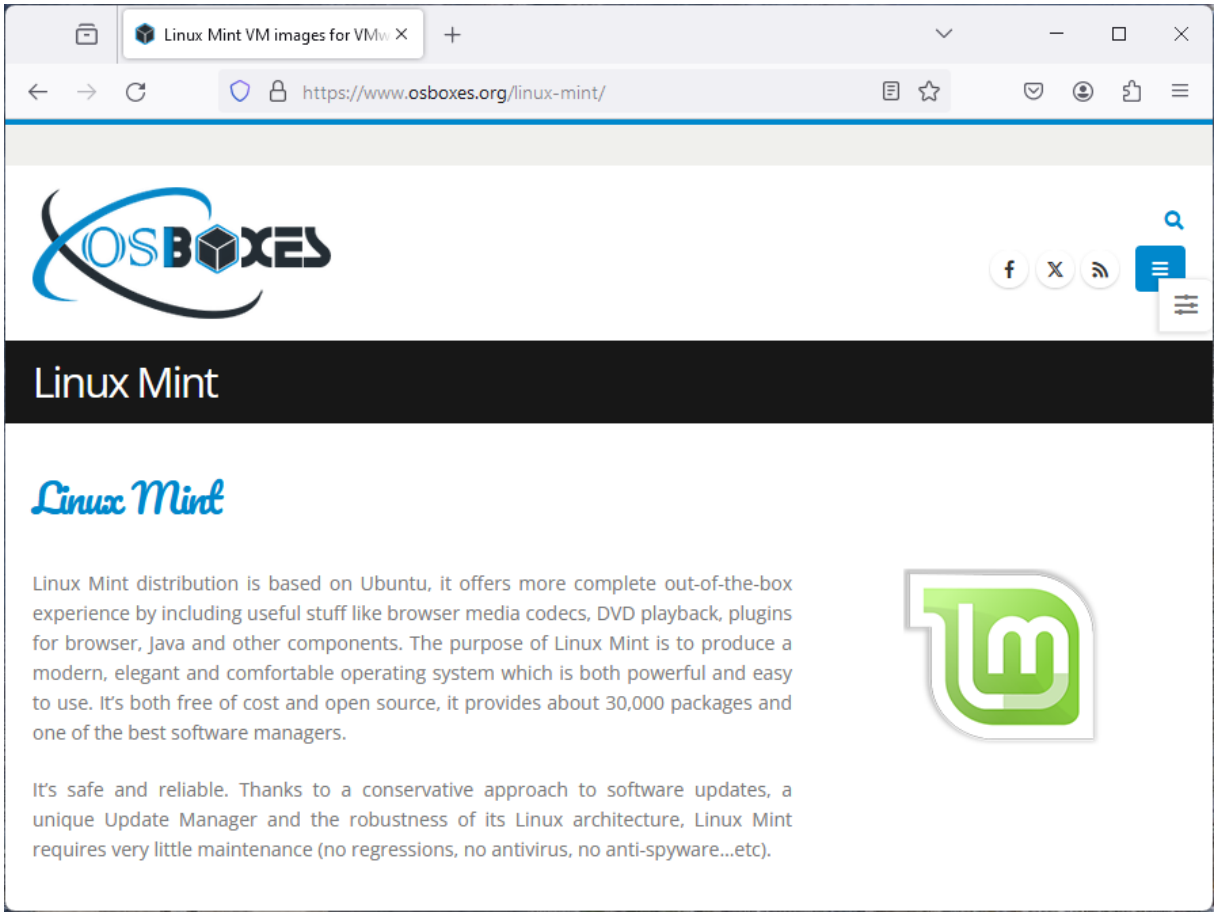


Figure 2.1.: The Linux Mint download page on the OSBoxes website.

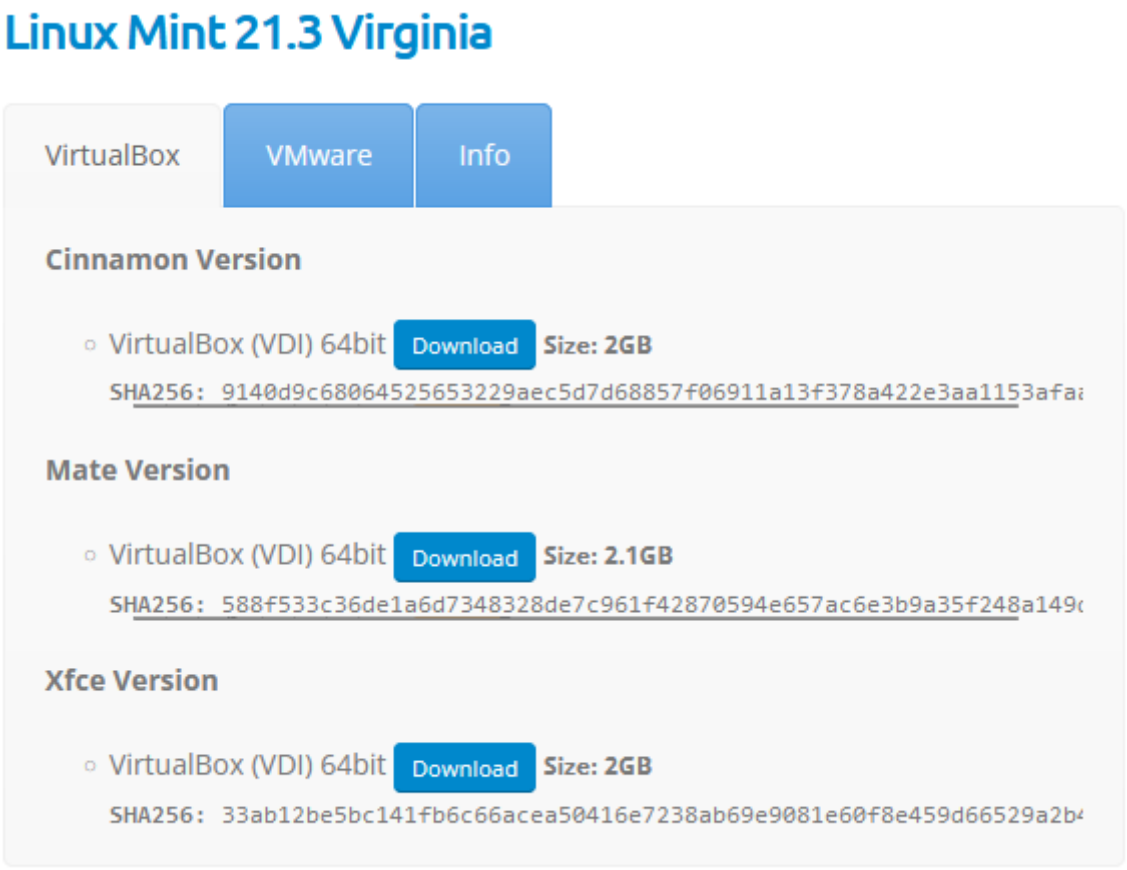


Figure 2.2.: The download section on the Linux Mint download page on the OSBoxes website.

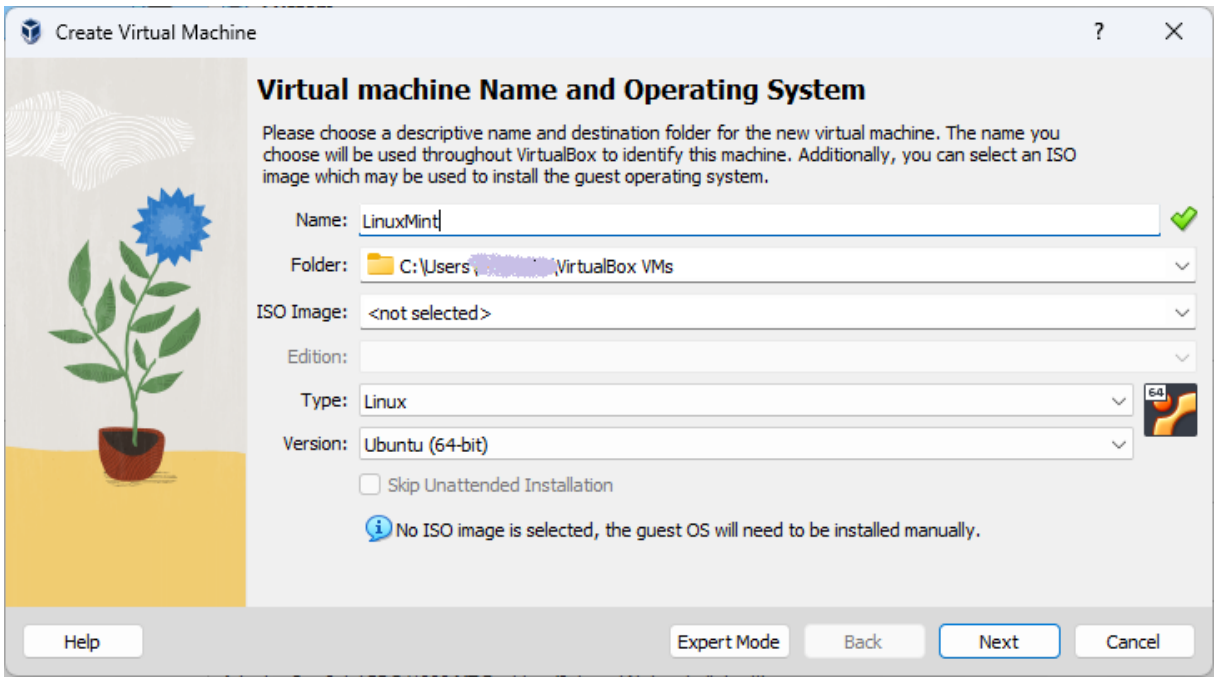


Figure 2.3.: The VirtualBox New VM dialog.

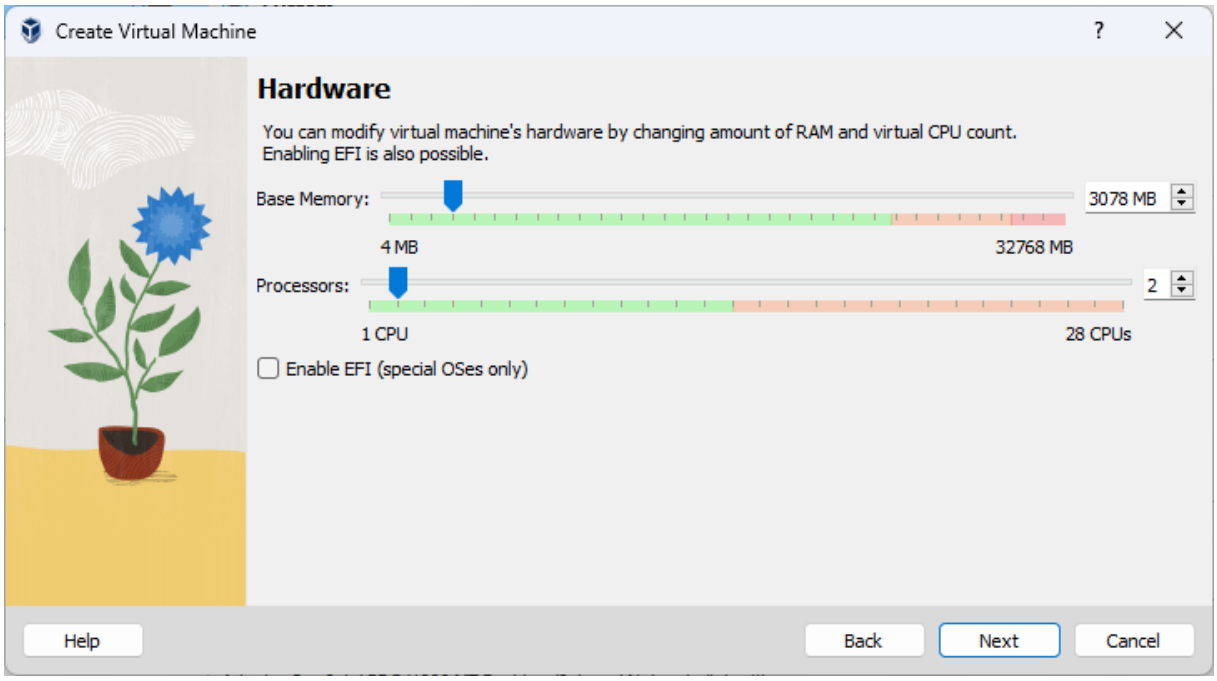


Figure 2.4.: Assigning memory and CPU cores to the virtual machine.

Click *Next* to proceed to the next step, selecting a Virtual Hard disk. Since we already have a .vdi file, we can use that as the hard disk for the virtual machine. Now is a good time to go to put the downloaded .vdi file in the correct location. By now, VirtualBox has created a folder for the virtual machine. All VMs are usually kept in a directory called *VirtualBox VMs* in your home directory. Inside that directory, there should now be a subdirectory with the name of the virtual machine. Move or copy the downloaded .vdi file into that directory.

Go back to the VirtualBox dialog and select *Use an existing virtual hard disk file*.

Click the folder icon on the right to open the *Hard Disk Selector dialog*. Here, click the Add button. The file dialog that opens will probably open in the correct directory and you should see the .vdi file. If not, navigate to the directory where you put the .vdi file. Select the file and click *Open*.

You're back in the Hard Disk Selector dialog. The .vdi file is shown under the category *Not Attached*. Ensure it's selected and click *Choose*. You're now back in the New Virtual Machine Wizard. Click *Next* to go to the final step showing a summary of the chosen settings. Click *Finish* to create the virtual machine.

In the main VirtualBox window, you should now see the new virtual machine in the list of VMs. It's ready to boot, but if you want to change some of the settings, you still can. For example, you could increase the video memory to the maximum value or enable 3D acceleration. Another useful change is to add a second network adapter attached to a Host-only Adapter or internal network. This will later allow you to create a virtual network with multiple VMs.

If you're happy with the settings, click the *Start* button to boot the virtual machine.

2.3. Logging in, initial configuration

After booting, you're greeted by the Linux Mint login screen. The default username is always *osboxes* and the password is *osboxes.org*. **Pay attention!** All osboxes-VMs are configured with a qwerty keyboard layout. Luckily, you can change the layout from the menu in the top right corner of the screen. For example, here we selected the *Belgian* layout.

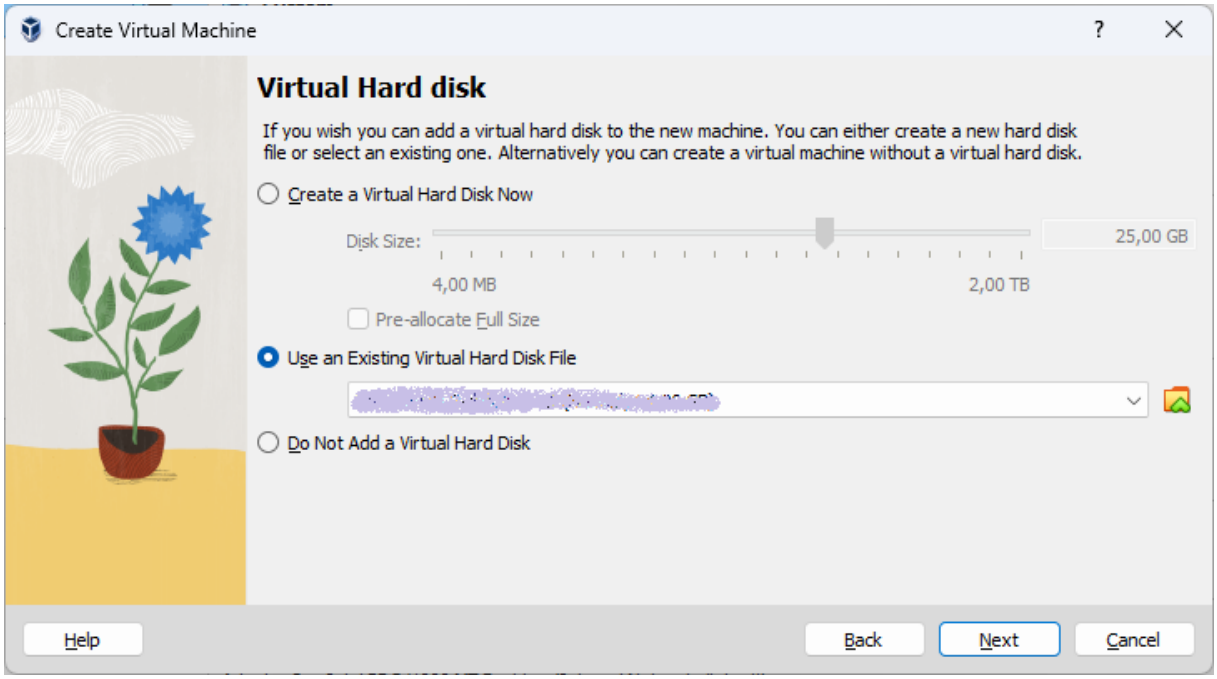


Figure 2.5.: Options for creating/selecting a virtual hard disk file.

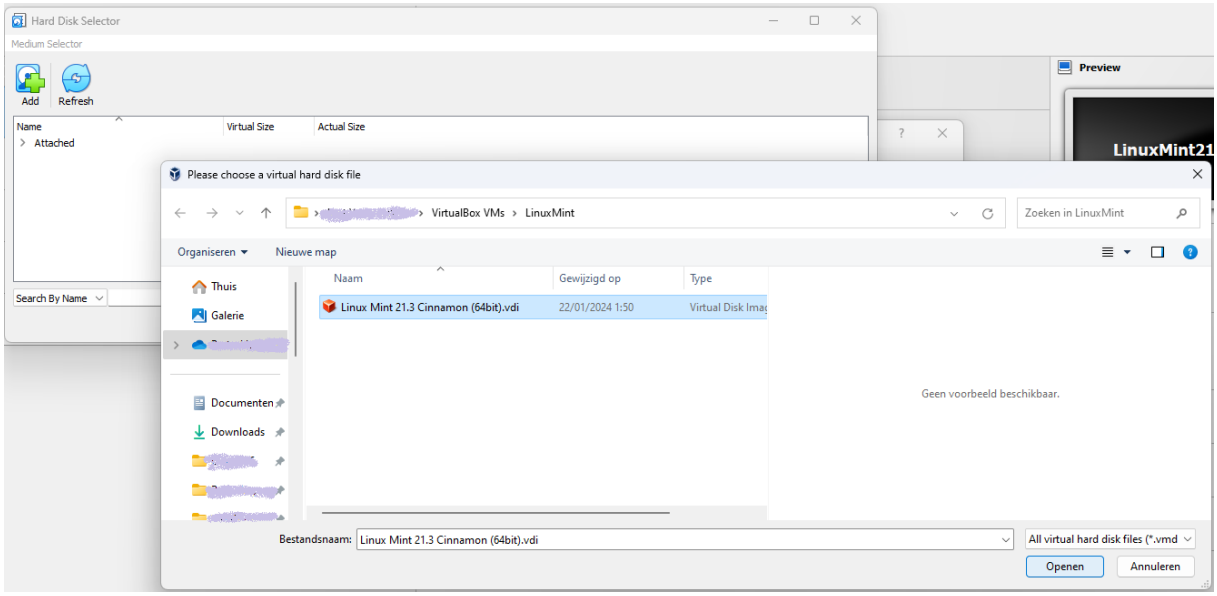


Figure 2.6.: The Hard Disk Selector dialog and selecting the .vdi file.

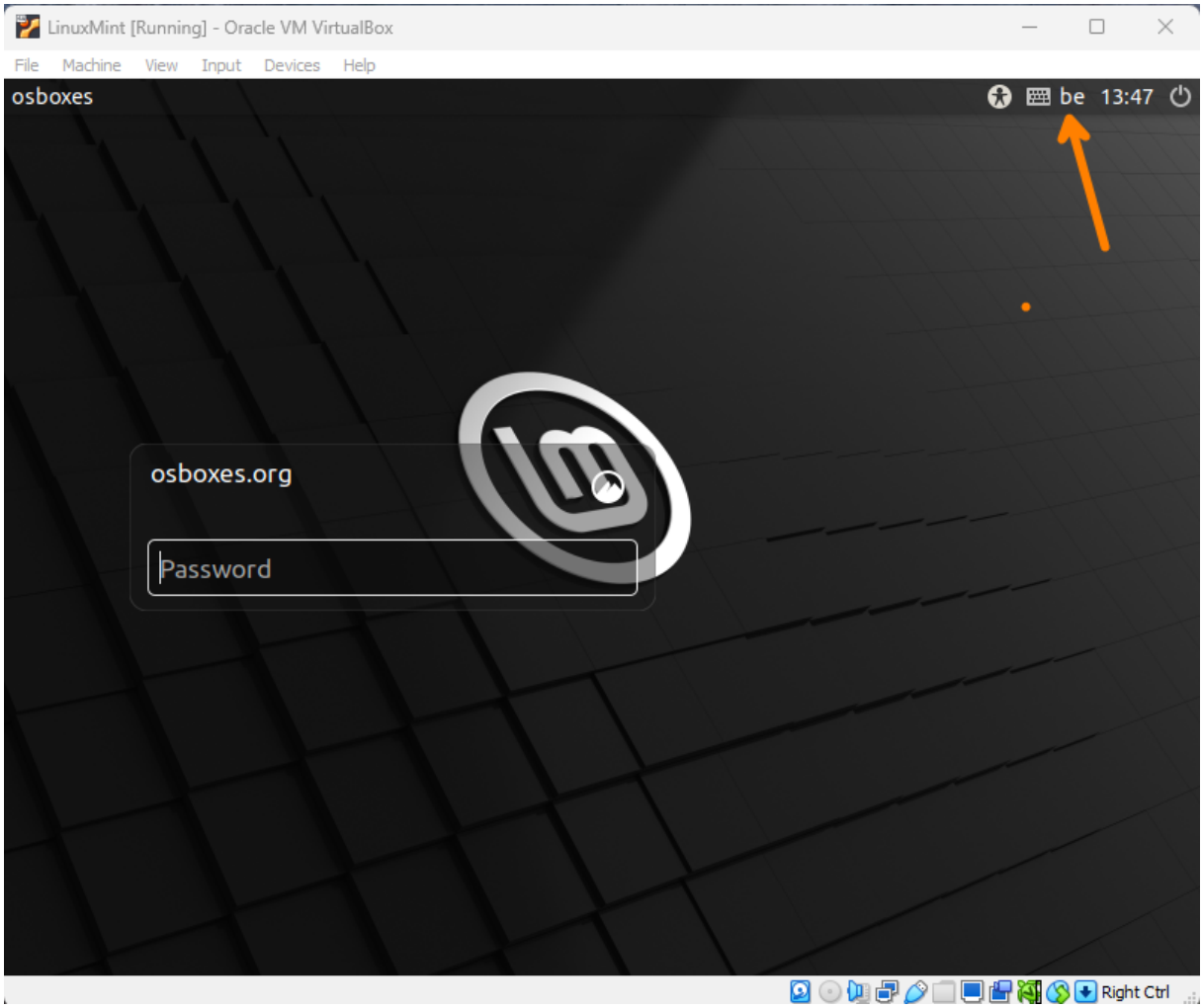


Figure 2.7.: The Linux Mint login screen with Belgian keyboard layout selected.

After logging in, you're presented with the Linux Mint desktop. Congratulations! You now have a virtual machine that you can use to learn Linux. You can start by exploring the desktop environment, the file manager, the web browser, the terminal, and the software manager. The big advantage of using a virtual machine is that you can experiment without the risk of breaking your computer. You can always start over by deleting the virtual machine and creating a new one.

To access the system settings, click the Linux Mint logo in the bottom left corner and select *System Settings* icon (indicated in the screenshot below). Here you can change the desktop background, the theme, the screensaver, the power settings, the keyboard layout, and much more.

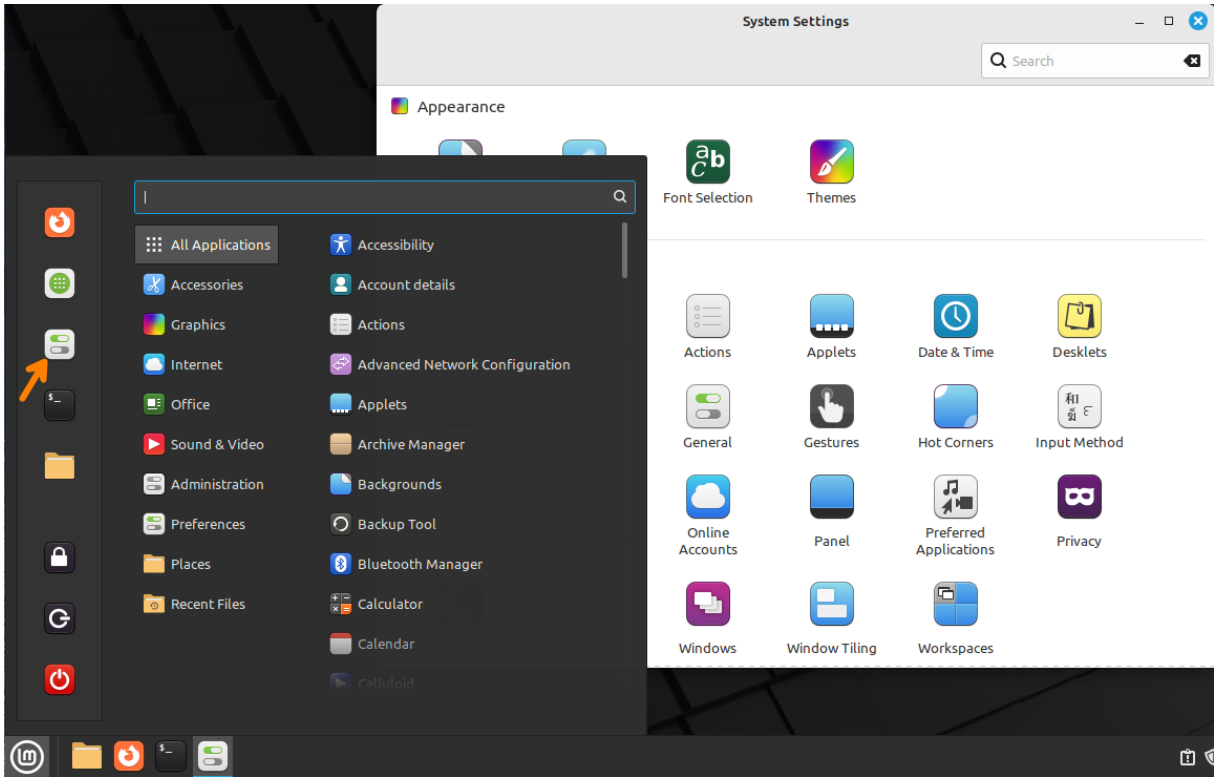


Figure 2.8.: The Linux Mint desktop with the System Settings icon highlighted by the orange arrow. The system settings app is running and partly covered by the start menu.

Feel free to use and configure the system as you like. We do have a few recommendations for you:

- **Disable the screensaver and automatic screen lock** (System Settings > Screensaver). This can be annoying when you're working in the VM and the screen locks after a few minutes of inactivity.

- If you have another **keyboard layout** than the default US QWERTY, change the layout to your preference and/or add alternative layouts (System Settings > Keyboard > Layouts).

- Optionally, **change your password** to one that you choose (System Settings > Account details > Password). If you're afraid of forgetting your password, here's a tip: open the VirtualBox VM Settings and go to General and then the Description tab. Here you can add a note with your password.

- In the VirtualBox window that contains your VM, open the **Devices menu** and select *Shared Clipboard* > *Bidirectional*. This will allow you to copy and paste text between your host system and the VM.

- After a few minutes, the *Update Manager* will be launched automatically. You can use this tool to **update the system** and install the latest security patches. Follow the instructions shown. If you're asked for a password, remember it's *osboxes.org*. After the updates are installed, you will probably be asked to restart the system. You can do this now or later.
- When you have configured your system to your liking, you can **create a snapshot of the VM**. This is a point-in-time backup of the VM that you can revert to if you mess up the system. To create a snapshot, go to the VirtualBox main window, ensure your Linux Mint VM is selected and click the settings icon right of the VM name. In the menu that pops up, select Snapshots.

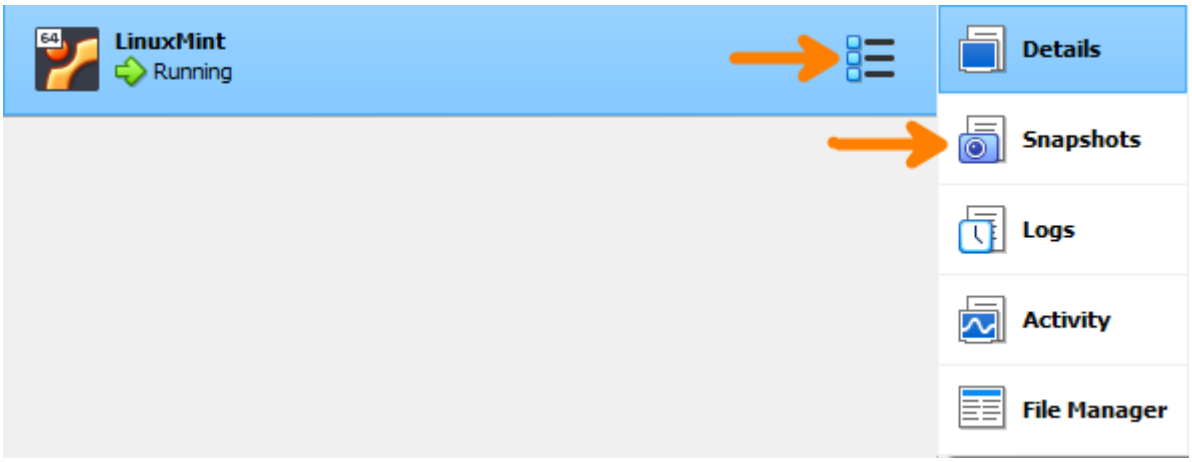


Figure 2.9.: How to create a snapshot of a VM in VirtualBox.

2.4. Shut down the VM

When you're done using the VM, you can shut it down. There are four good ways to stop the VM and one particularly bad one.

1. You can click the *Linux Mint Start Menu* button in the bottom left corner, then the *Shut Down* button (the red icon on the bottom). In the dialog that appears, select *Shut Down*.
2. You can open a terminal and enter the command `poweroff`.
3. You can click the close button (the X in the top right corner of the VM window). In the dialog that appears, select *Send the shutdown signal*. Your VM should show the dialog as in the previous method where you can select *Shut Down*.
4. You can click the close button of the VM window and select *Save the machine state*. This is basically the equivalent of closing your laptop lid. The next time you start the VM, it will resume from where you left off.
5. You can click the close button of the VM window and select *Power off the machine*. This is the bad way to shut down the VM because it's the equivalent of pulling the power plug of a physical computer. It's not recommended to do this, since it may cause corruption of the file system when a write operation was in progress at the time of the power off.

2.5. Exercises

1. Try to start the most common applications that you expect to be installed on a basic system, e.g. the web browser, a text editor, the file manager, a word processor, spreadsheet application, drawing app, etc.

2. Try use the software manager to install a new application, e.g. Terminator (an alternative for the Terminal application), Git (a revision control system), Shellcheck (a tool that helps you write better shell scripts), Vim (Vi iMproved, a powerful text editor), Visual Studio Code, etc.

Part II.

Organising users

3. standard file permissions

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/, Bert Van Vreckem, https://github.com/bertvv/)

This chapter contains details about basic file security through *file ownership* and *file permissions*.

3.1. file ownership

3.1.1. user owner and group owner

The *users* and *groups* of a system can be locally managed in `/etc/passwd` and `/etc/group`, or they can be in a NIS, LDAP, or Samba domain. These users and groups can *own* files. Actually, every file has a *user owner* and a *group owner*, as can be seen in the following example.

```
student@linux:~/owners$ ls -lh
total 636K
-rw-r--r--. 1 student snooker  1.1K Apr  8 18:47 data.odt
-rw-r--r--. 1 student student  626K Apr  8 18:46 file1
-rw-r--r--. 1 student tennis    185 Apr  8 18:46 file2
-rw-rw-r--. 1 root root           0 Apr  8 18:47 stuff.txt
```

User `student` owns three files: `file1` has `student` as *user owner* and has the group `student` as *group owner*, `data.odt` is *group owned* by the group `snooker`, `file2` by the group `tennis`.

The last file is called `stuff.txt` and is owned by the `root` user and the `root` group.

3.1.2. chgrp

You can change the group owner of a file using the `chgrp` command. You must have root privileges to do this.

```
root@linux:/home/student/owners# ls -l file2
-rw-r--r--. 1 root tennis 185 Apr  8 18:46 file2
root@linux:/home/student/owners# chgrp snooker file2
root@linux:/home/student/owners# ls -l file2
-rw-r--r--. 1 root snooker 185 Apr  8 18:46 file2
root@linux:/home/student/owners#
```

3.1.3. chown

The user owner of a file can be changed with `chown` command. You must have root privileges to do this. In the following example, the user owner of `file2` is changed from `root` to `student`.

```
root@linux:/home/student# ls -l FileForStudent
-rw-r--r-- 1 root student 0 2008-08-06 14:11 FileForStudent
root@linux:/home/student# chown student FileForStudent
root@linux:/home/student# ls -l FileForStudent
-rw-r--r-- 1 student student 0 2008-08-06 14:11 FileForStudent
```

You can also use `chown user:group` to change both the user owner and the group owner.

```
root@linux:/home/student# ls -l FileForStudent
-rw-r--r-- 1 student student 0 2008-08-06 14:11 FileForStudent
root@linux:/home/student# chown root:project42 FileForStudent
root@linux:/home/student# ls -l FileForStudent
-rw-r--r-- 1 root project42 0 2008-08-06 14:11 FileForStudent
```

3.2. list of special files

When you use `ls -l`, for each file you can see ten characters before the user and group owner. The first character tells us the type of file. Regular files get a `-`, directories get a `d`, symbolic links are shown with an `l`, pipes get a `p`, character devices a `c`, block devices a `b`, and sockets an `s`.

first character	file type
-	normal file
d	directory
l	symbolic link
p	named pipe
b	block device
c	character device
s	socket

Below an example of a character device (the console) and a block device (the hard disk).

```
student@linux:~$ ls -l /dev/console /dev/sda
crw--w---- 1 root tty  5, 1 Mar  8 08:32 /dev/console
brw-rw---- 1 root disk 8, 0 Mar  8 08:32 /dev/sda
```

And here you can see a directory, a regular file and a symbolic link.

```
student@linux:~$ ls -ld /etc /etc/hosts /etc/os-release
drwxr-xr-x 81 root root 4096 Mar  8 08:32 /etc
-rw-r--r--  1 root root  186 Feb 26 14:58 /etc/hosts
lrwxrwxrwx  1 root root   21 Dec  9 21:08 /etc/os-release -> ../usr/lib/os-
release
```

3.3. permissions

3.3.1. rwx

The nine characters following the file type denote the permissions in three triplets. A permission can be `r` for **r**ead access, `w` for **w**rite access, and `x` for e**x**ecute. You need the `r` permission to list (ls) the contents of a directory. You need the `x` permission to enter (cd) a directory. You need the `w` permission to create files in or remove files from a directory.

permission	on a file	on a directory
read	read file contents (`cat`)	read directory contents (`ls`)
write	change file contents	create/delete files (`touch`,`rm`)
e**x**ecute	execute the file	enter the directory (`cd`)

3.3.2. three sets of rwx

We already know that the output of `ls -l` starts with ten characters for each file. This example shows a regular file (because the first character is a -).

```
student@linux:~/test$ ls -l proc42.sh
-rwxr-xr--  1 student proj  984 Feb  6 12:01 proc42.sh
```

Below is a table describing the function of all ten characters.

position	characters	function
1	`-`	file type
2-4	`rwx`	permissions for the *user owner*
5-7	`r-x`	permissions for the *group owner*
8-10	`r--`	permissions for *others*

When you are the *user owner* of a file, then the *user owner permissions* apply to you. The rest of the permissions have no influence on your access to the file.

When you belong to the *group* that is the *group owner* of a file, then the *group owner permissions* apply to you. The rest of the permissions have no influence on your access to the file.

When you are not the *user owner* of a file and you do not belong to the *group owner*, then the *others permissions* apply to you. The rest of the permissions have no influence on your access to the file.

3.3.3. permission examples

Some example combinations on files and directories are seen in this example. The name of the file explains the permissions.

```
student@linux:~/perms$ ls -lh
total 12K
drwxr-xr-x 2 student student 4.0K 2007-02-07 22:26 AllEnter_UserCreateDelete
-rwxrwxrwx 1 student student    0 2007-02-07 22:21 EveryoneFullControl.txt
-r--r----- 1 student student    0 2007-02-07 22:21 OnlyOwnersRead.txt
-rwxrwx--- 1 student student    0 2007-02-07 22:21 OwnersAll_RestNothing.txt
dr-xr-x--- 2 student student 4.0K 2007-02-07 22:25 UserAndGroupEnter
dr-x------ 2 student student 4.0K 2007-02-07 22:25 OnlyUserEnter
```

To summarise, the first `rwx` triplet represents the permissions for the *user owner*. The second triplet corresponds to the *group owner*; it specifies permissions for all members of that group. The third triplet defines permissions for all *other* users that are not the *user owner* and are not a member of the *group owner*. The `root` user ignores all restrictions and can do anything with any file.

3.3.4. setting permissions with symbolic notation

Permissions can be changed with `chmod MODE FILE ...`. You need to be the owner of the file to do this. The first example gives (`+`) the *user owner* (`u`) execute (`x`) permissions.

```
student@linux:~/perms$ ls -l permissions.txt
-rw-r--r-- 1 student student 0 2007-02-07 22:34 permissions.txt
student@linux:~/perms$ chmod u+x permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwxr--r-- 1 student student 0 2007-02-07 22:34 permissions.txt
```

This example removes (`-`) the group owner's (`g`) read (`r`) permission.

```
student@linux:~/perms$ chmod g-r permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwx---r-- 1 student student 0 2007-02-07 22:34 permissions.txt
```

This example removes (`-`) the other's (`o`) read (`r`) permission.

```
student@linux:~/perms$ chmod o-r permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwx------ 1 student student 0 2007-02-07 22:34 permissions.txt
```

This example gives (`+`) all (`a`) of them the write (`w`) permission.

```
student@linux:~/perms$ chmod a+w permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwx-w--w- 1 student student 0 2007-02-07 22:34 permissions.txt
```

You don't even have to type the `a`.

```
student@linux:~/perms$ chmod +x permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwx-wx-wx 1 student student 0 2007-02-07 22:34 permissions.txt
```

You can also set explicit permissions with `=`.

```
student@linux:~/perms$ chmod u=rw permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rw--wx-wx 1 student student 0 2007-02-07 22:34 permissions.txt
```

Feel free to make any kind of combination, separating them with a comma. Remark that spaces are **not** allowed!

```
student@linux:~/perms$ chmod u=rw,g=rw,o=r permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 student student 0 2007-02-07 22:34 permissions.txt
```

Even fishy combinations are accepted by `chmod`.

```
student@linux:~/perms$ chmod u=rwx,ug+rw,o=r permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwxrw-r-- 1 student student 0 2007-02-07 22:34 permissions.txt
```

Summarized, in order to change permissions with `chmod` using symbolic notation:

- first specify who the permissions are for: `u` for the user owner, `g` for the group owner, `o` for others, and `a` for all. `a` is the default and can be omitted.
- then specify the operation: `+` to add permissions, `-` to remove permissions, and `=` to set permissions.
- finally specify the permission(s): `r` for read, `w` for write, and `x` for execute.
- multiple operations can be combined with a comma (no spaces!)

3.3.5. setting permissions with octal notation

Most Unix administrators will use the "old school" octal system to talk about and set permissions. Consider the triplet to be a binary number with 0 indicating the permission is not set and 1 indicating the permission is set. You then have $2^3 = 8$ possible combinations, hence the name *octal*. You can then convert the binary number to an octal number, equating `r` to 4, `w` to 2, and `x` to 1.

permission	binary	octal
`---`	000	0
`--x`	001	1
`-w-`	010	2
`-wx`	011	3
`r--`	100	4
`r-x`	101	5
`rw-`	110	6
`rwx`	111	7

Since we have three triplets, we can use three octal digits to represent the permissions. This makes 777 equal to `rwxrwxrwx` and by the same logic, 654 mean `rw-r-xr--`. The `chmod` command will accept these numbers.

```
student@linux:~/perms$ chmod 777 permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwxrwxrwx 1 student student 0 2007-02-07 22:34 permissions.txt
student@linux:~/perms$ chmod 664 permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 student student 0 2007-02-07 22:34 permissions.txt
student@linux:~/perms$ chmod 750 permissions.txt
student@linux:~/perms$ ls -l permissions.txt
-rwxr-x--- 1 student student 0 2007-02-07 22:34 permissions.txt
```

Remark that in practice, some combinations will never occur:

- The permissions of a user will never be smaller than the permissions of the group owner or others. Consequently, the digits will always be in descending order.
- Setting the write or execute permission without read access is useless. Consequently, you will never use 1, 2, or 3 in an octal permission code

- A directory will always have the read and execute permission set or unset together. It is useless to allow a user to read the directory contents, but not let them `cd` into that directory. Allowing `cd` without read access is also useless. The permission code for a directory will therefore always be odd.

Here's a little tip: you can print the permissions of a file in either octal or symbolic notation with the `stat` command (check the man page of `stat` to see how this works).

```
[student@linux ~]$ stat -c '%A %a' /etc/passwd
-rw-r--r-- 644
[student@linux ~]$ stat -c '%A %a' /etc/shadow
---------- 0
[student@linux ~]$ stat -c '%A %a' /bin/ls
-rwxr-xr-x 755
```

3.3.6. umask

When creating a file or directory, a set of default permissions are applied. These default permissions are determined by the `umask` value. The `umask` specifies permissions that you do not want set on by default. You can display the `umask` with the `umask` command.

```
[student@linux ~]$ umask
0002
[student@linux ~]$ touch test
[student@linux ~]$ ls -l test
-rw-rw-r--  1 student student    0 Jul 24 06:03 test
[student@linux ~]$
```

As you can also see, the file is also not executable by default. This is a general security feature among Unixes; newly created files are never executable by default. You have to explicitly do a `chmod +x` to make a file executable. This also means that the 1 bit in the `umask` has no meaning. A `umask` value of 0022 has the same effect as 0033.

In practice, you will only use umask values:

- 0: don't take away any permissions
- 2: take away write permissions
- 7: take away all permissions

You can set the `umask` value to a new value with the `umask` command. The `umask` value is a four-digit octal number. The first digit is for special permissions (and is always zero), the second for the user permissions (is in practice always 0, since there is no use in taking away the user's permissions), the third for the group owner (sometimes 0, but usually 2 or 7), and the last for others (usually 2 or 7, 0 is very uncommon and can be considered to be a security risk).

The `umask` value is subtracted from 777 to get the default permissions and in the case of a file, the execute bit is removed.

```
[student@linux ~]$ umask 0002
[student@linux ~]$ touch file0002
[student@linux ~]$ mkdir dir0002
[student@linux ~]$ ls -ld *0002
drwxrwxr-x. 2 student student 6 Mar  8 10:48 dir0002
-rw-rw-r--. 1 student student 0 Mar  8 10:47 file0002
[student@linux ~]$ umask 0027
[student@linux ~]$ touch file0027
[student@linux ~]$ mkdir dir0027
```

```
[student@linux ~]$ ls -ld *0027
drwxr-x---. 2 student student 6 Mar  8 10:48 dir0027
-rw-r-----. 1 student student 0 Mar  8 10:48 file0027
[student@linux ~]$ umask 0077
[student@linux ~]$ touch file0077
[student@linux ~]$ mkdir dir0077
[student@linux ~]$ ls -ld *0077
drwx------. 2 student student 6 Mar  8 10:51 dir0077
-rw-------. 1 student student 0 Mar  8 10:51 file0077
```

3.3.7. mkdir -m

When creating directories with `mkdir` you can use the `-m` option to set the `mode`. This example explains.

```
student@linux~$ mkdir -m 700 MyDir
student@linux~$ mkdir -m 777 Public
student@linux~$ ls -dl MyDir/ Public/
drwx------ 2 student student 4096 2011-10-16 19:16 MyDir/
drwxrwxrwx 2 student student 4096 2011-10-16 19:16 Public/
```

3.3.8. cp -p

To preserve permissions and time stamps from source files, use `cp -p`.

```
student@linux:~/perms$ cp file* cp
student@linux:~/perms$ cp -p file* cpp
student@linux:~/perms$ ll *
-rwx------ 1 student student    0 2008-08-25 13:26 file33
-rwxr-x--- 1 student student    0 2008-08-25 13:26 file42

cp:
total 0
-rwx------ 1 student student 0 2008-08-25 13:34 file33
-rwxr-x--- 1 student student 0 2008-08-25 13:34 file42

cpp:
total 0
-rwx------ 1 student student 0 2008-08-25 13:26 file33
-rwxr-x--- 1 student student 0 2008-08-25 13:26 file42
```

3.4. practice: standard file permissions

1. As normal user, create a directory `~/permissions`. Create a file owned by yourself in there.
2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?
3. As root, create a file in the users `~/permissions` directory.
4. As normal user, look at who owns this file created by root.
5. Change the ownership of all files in `~/permissions` to yourself.
6. Delete the file created by root. Is this possible?

7. With chmod, is 770 the same as `rwxrwx---`?
8. With chmod, is 664 the same as `r-xr-xr--`?
9. With chmod, is 400 the same as `r--------`?
10. With chmod, is 734 the same as `rwxr-xr--`?
11. Display the umask value in octal and in symbolic form.
12. Set the umask to 0077, but use the symbolic format to set it. Verify that this works.
13. Create a file as root, give only read to others. Can a normal user read this file? Test writing to this file with `vi` or `nano`.
14. Create a file as a normal user, take away all permissions for the group owner and others. Can you still read the file? Can root read the file? Can root write to the file?
15. Create a directory that belongs to group `users`, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

3.5. solution: standard file permissions

1. As normal user, create a directory `~/permissions`. Create a file owned by yourself in there.

```
[student@linux ~]$ mkdir permissions
[student@linux ~]$ touch permissions/myfile.txt
[student@linux ~]$ ls -l permissions/
total 0
-rw-r--r--. 1 student student 0 Mar  8 10:59 myfile.txt
```

2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?

```
[student@linux ~]$ ls -l /etc/hosts
-rw-r--r--. 1 root root 174 Feb 26 15:05 /etc/hosts
[student@linux ~]$ cp /etc/hosts ~/permissions/
[student@linux ~]$ ls -l permissions/hosts
-rw-r--r--. 1 student student 174 Mar  8 11:00 permissions/hosts
```

   The copy is owned by you.

3. As root, create a file in the users `~/permissions` directory.

```
[student@linux ~]$ sudo touch permissions/rootfile.txt
[sudo] password for student:
```

4. As normal user, look at who owns this file created by root.

```
[student@linux ~]$ ls -l permissions/*.txt
-rw-r--r--. 1 student student 0 Mar  8 10:59 permissions/myfile.txt
-rw-r--r--. 1 root    root    0 Mar  8 11:02 permissions/rootfile.txt
```

   The file created by root is owned by root.

5. Change the ownership of all files in ~/permissions to yourself.

```
[student@linux ~]$ chown student ~/permissions/*
chown: changing ownership of '/home/student/permissions/rootfile.txt': Operation not p
```

   You cannot become owner of the file that belongs to root. Root must change the ownership.

6. Delete the file created by root. Is this possible?

```
[student@linux ~]$ rm ~/permissions/rootfile.txt
rm: remove write-protected regular empty file '/home/student/permissions/rootfile.txt'
[student@linux ~]$ ls -l permissions/*.txt
-rw-r--r--. 1 student student 0 Mar  8 10:59 permissions/myfile.txt
```

You can delete the file since you have write permission on the directory!

7. With chmod, is 770 the same as `rwxrwx---`?

   yes

8. With chmod, is 664 the same as `r-xr-xr--`?

   no, `rw-rw-r--` is 664 and `r-xr-xr--` is 774

9. With chmod, is 400 the same as `r--------`?

   yes

10. With chmod, is 734 the same as `rwxr-xr--`?

    no, `rwxr-xr--` is 754 and `rwx-wxr--` is 734

11. Display the umask in octal and in symbolic form.

    `umask` and `umask -S`

12. Set the umask to 0077, but use the symbolic format to set it. Verify that this works.

    ```
    [student@linux ~]$ umask -S u=rwx,go=
    u=rwx,g=,o=
    [student@linux ~]$ umask
    0077
    ```

13. Create a file as root, give only read to others. Can a normal user read this file? Test writing to this file with `vi` or `nano`.

    ```
    [student@linux ~]$ sudo vi permissions/rootfile.txt
    [student@linux ~]$ sudo chmod 644 permissions/rootfile.txt
    [student@linux ~]$ ls -l permissions/*.txt
    -rw-r--r--. 1 student student 0 Mar  8 10:59 permissions/myfile.txt
    -rw-r--r--. 1 root    root    6 Mar  8 13:53 permissions/rootfile.txt
    [student@linux ~]$ cat permissions/rootfile.txt
    hello
    [student@linux ~]$ echo " world" >> permissions/rootfile.txt
    -bash: permissions/rootfile.txt: Permission denied
    ```

    Yes, a normal user can read the file, but not write to it.

14. Create a file as a normal user, take away all permissions for the group and others. Can you still read the file? Can root read the file? Can root write to the file?

    ```
    [student@linux ~]$ vi permissions/privatefile.txt
    ... (editing the file) ...
    [student@linux ~]$ cat permissions/privatefile.txt
    hello
    [student@linux ~]$ chmod 600 permissions/privatefile.txt
    [student@linux ~]$ ls -l permissions/privatefile.txt
    -rw-------. 1 student student 0 Mar  8 16:06 permissions/privatefile.txt
    [student@linux ~]$ cat permissions/privatefile.txt
    hello
    ```

    Of course, the owner can still read (and write to) the file.

```
[student@linux ~]$ sudo vi permissions/privatefile.txt
[sudo] password for student:
... (editing the file) ...
[student@linux ~]$ cat permissions/privatefile.txt
hello world
```

Root can read and write to the file. In fact, root ignores all file permissions and can do anything with any file.

15. Create a directory `shared/` that belongs to group `users`, where every member of that group can read and write to files, and create files.

```
[student@linux ~]$ mkdir shared
[student@linux ~]$ sudo chgrp users shared
[student@linux ~]$ chmod 775 shared/
[student@linux ~]$ ls -ld shared/
drwxrwxr-x. 2 student users 6 Mar  8 18:26 shared/
```

4. advanced file permissions

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

4.1. sticky bit on directory

You can set the `sticky bit` on a directory to prevent users from removing files that they do not own as a user owner. The sticky bit is displayed at the same location as the x permission for others. The sticky bit is represented by a `t` (meaning x is also there) or a `T` (when there is no x for others).

```
root@linux:~# mkdir /project55
root@linux:~# ls -ld /project55
drwxr-xr-x  2 root root 4096 Feb  7 17:38 /project55
root@linux:~# chmod +t /project55/
root@linux:~# ls -ld /project55
drwxr-xr-t  2 root root 4096 Feb  7 17:38 /project55
root@linux:~#
```

The `sticky bit` can also be set with octal permissions, it is binary `1` in the first of four triplets.

```
root@linux:~# chmod 1775 /project55/
root@linux:~# ls -ld /project55
drwxrwxr-t  2 root root 4096 Feb  7 17:38 /project55
root@linux:~#
```

You will typically find the `sticky bit` on the `/tmp` directory.

```
root@linux:~# ls -ld /tmp
drwxrwxrwt 6 root root 4096 2009-06-04 19:02 /tmp
```

4.2. setgid bit on directory

`setgid` can be used on directories to make sure that all files inside the directory are owned by the group owner of the directory. The `setgid` bit is displayed at the same location as the x permission for group owner. The `setgid` bit is represented by an `s` (meaning x is also there) or a `S` (when there is no x for the group owner). As this example shows, even though `root` does not belong to the group proj55, the files created by root in /project55 will belong to proj55 since the `setgid` is set.

```
root@linux:~# groupadd proj55
root@linux:~# chown root:proj55 /project55/
root@linux:~# chmod 2775 /project55/
root@linux:~# touch /project55/fromroot.txt
root@linux:~# ls -ld /project55/
drwxrwsr-x  2 root proj55 4096 Feb  7 17:45 /project55/
root@linux:~# ls -l /project55/
total 4
-rw-r--r--  1 root proj55 0 Feb  7 17:45 fromroot.txt
root@linux:~#
```

You can use the `find` command to find all `setgid` directories.

```
student@linux:~$ find / -type d -perm -2000 2> /dev/null
/var/log/mysql
/var/log/news
/var/local
...
```

4.3. setgid and setuid on regular files

These two permissions cause an executable file to be executed with the permissions of the `file owner` instead of the `executing owner`. This means that if any user executes a program that belongs to the `root user`, and the `setuid` bit is set on that program, then the program runs as `root`. This can be dangerous, but sometimes this is good for security.

Take the example of passwords; they are stored in `/etc/shadow` which is only readable by `root`. (The `root` user never needs permissions anyway.)

```
root@linux:~# ls -l /etc/shadow
-r--------  1 root root 1260 Jan 21 07:49 /etc/shadow
```

Changing your password requires an update of this file, so how can normal non-root users do this? Let's take a look at the permissions on the `/usr/bin/passwd`.

```
root@linux:~# ls -l /usr/bin/passwd
-r-s--x--x  1 root root 21200 Jun 17  2005 /usr/bin/passwd
```

When running the `passwd` program, you are executing it with `root` credentials.

You can use the `find` command to find all `setuid` programs.

```
student@linux:~$ find /usr/bin -type f -perm -04000
/usr/bin/arping
/usr/bin/kgrantpty
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/fping6
/usr/bin/passwd
/usr/bin/gpasswd
...
```

In most cases, setting the `setuid` bit on executables is sufficient. Setting the `setgid` bit will result in these programs to run with the credentials of their group owner.

4.4. setuid on sudo

The `sudo` binary has the `setuid` bit set, so any user can run it with the effective userid of root.

```
student@linux:~$ ls -l $(which sudo)
---s--x--x. 1 root root 123832 Oct  7  2013 /usr/bin/sudo
student@linux:~$
```

4.5. practice: sticky, setuid and setgid bits

1a. Set up a directory, owned by the group sports.

1b. Members of the sports group should be able to create files in this directory.

1c. All files created in this directory should be group-owned by the sports group.

1d. Users should be able to delete only their own user-owned files.

1e. Test that this works!

2. Verify the permissions on `/usr/bin/passwd`. Remove the `setuid`, then try changing your password as a normal user. Reset the permissions back and try again.

3. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

4.6. solution: sticky, setuid and setgid bits

1a. Set up a directory, owned by the group sports.

```
groupadd sports

mkdir /home/sports

chown root:sports /home/sports
```

1b. Members of the sports group should be able to create files in this directory.

```
chmod 770 /home/sports
```

1c. All files created in this directory should be group-owned by the sports group.

```
chmod 2770 /home/sports
```

1d. Users should be able to delete only their own user-owned files.

```
chmod +t /home/sports
```

1e. Test that this works!

Log in with different users (group members and others and root), create files and watch the permissions. Try changing and deleting files...

2. Verify the permissions on `/usr/bin/passwd`. Remove the `setuid`, then try changing your password as a normal user. Reset the permissions back and try again.

```
root@linux:~# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
root@linux:~# chmod 755 /usr/bin/passwd
root@linux:~# ls -l /usr/bin/passwd
-rwxr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
```

A normal user cannot change password now.

```
root@linux:~# chmod 4755 /usr/bin/passwd
root@linux:~# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31704 2009-11-14 15:41 /usr/bin/passwd
```

3. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

```
student@linux:~$ sudo su -
[sudo] password for paul:
root@linux:~# mkdir attr
root@linux:~# cd attr/
root@linux:~/attr# touch file42
root@linux:~/attr# lsattr
---------------- ./file42
root@linux:~/attr# chattr +i file42
root@linux:~/attr# lsattr
----i----------- ./file42
root@linux:~/attr# rm -rf file42
rm: cannot remove `file42': Operation not permitted
root@linux:~/attr# chattr -i file42
root@linux:~/attr# rm -rf file42
root@linux:~/attr#
```

5. introduction to users

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This little chapter will teach you how to identify your user account on a Unix computer using commands like `who am i`, `id`, and more.

In a second part you will learn how to become another user with the `su` command.

And you will learn how to run a program as another user with `sudo`.

5.1. whoami

The `whoami` command tells you your username.

```
[student@linux ~]$ whoami
paul
[student@linux ~]$
```

5.2. who

The `who` command will give you information about who is logged on the system.

```
[student@linux ~]$ who
root     pts/0        2014-10-10 23:07 (10.104.33.101)
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
laura    pts/2        2014-10-10 23:34 (10.104.33.96)
tania    pts/3        2014-10-10 23:39 (10.104.33.91)
[student@linux ~]$
```

5.3. who am i

With `who am i` the `who` command will display only the line pointing to your current session.

```
[student@linux ~]$ who am i
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
[student@linux ~]$
```

5.4. w

The `w` command shows you who is logged on and what they are doing.

```
[student@linux ~]$ w
 23:34:07 up 31 min,  2 users,  load average: 0.00, 0.01, 0.02
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0     23:07   15.00s  0.01s  0.01s top
paul     pts/1     23:30    7.00s  0.00s  0.00s w
[student@linux ~]$
```

5.5. id

The `id` command will give you your user id, primary group id, and a list of the groups that you belong to.

```
student@linux:~$ id
uid=1000(paul) gid=1000(paul) groups=1000(paul)
```

On RHEL/CentOS you will also get `SELinux` context information with this command.

```
[root@linux ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r\
:unconfined_t:s0-s0:c0.c1023
```

5.6. su to another user

The `su` command allows a user to run a shell as another user.

```
laura@linux:~$ su tania
Password:
tania@linux:/home/laura$
```

5.7. su to root

Yes you can also `su` to become `root`, when you know the `root password`.

```
laura@linux:~$ su root
Password:
root@linux:/home/laura#
```

5.8. su as root

You need to know the password of the user you want to substitute to, unless your are logged in as `root`. The `root` user can become any existing user without knowing that user's password.

```
root@linux:~# id
uid=0(root) gid=0(root) groups=0(root)
root@linux:~# su - valentina
valentina@linux:~$
```

5.9. su - $username

By default, the `su` command maintains the same shell environment. To become another user and also get the target user's environment, issue the `su -` command followed by the target username.

```
root@linux:~# su laura
laura@linux:/root$ exit
exit
root@linux:~# su - laura
laura@linux:~$ pwd
/home/laura
```

5.10. su -

When no username is provided to `su` or `su -`, the command will assume `root` is the target.

```
tania@linux:~$ su -
Password:
root@linux:~#
```

5.11. run a program as another user

The sudo program allows a user to start a program with the credentials of another user. Before this works, the system administrator has to set up the `/etc/sudoers` file. This can be useful to delegate administrative tasks to another user (without giving the root password).

The screenshot below shows the usage of `sudo`. User `paul` received the right to run `useradd` with the credentials of `root`. This allows `paul` to create new users on the system without becoming `root` and without knowing the `root password`.

First the command fails for `paul`.

```
student@linux:~$ /usr/sbin/useradd -m valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

But with `sudo` it works.

```
student@linux:~$ sudo /usr/sbin/useradd -m valentina
[sudo] password for paul:
student@linux:~$
```

5.12. visudo

Check the man page of `visudo` before playing with the `/etc/sudoers` file. Editing the `sudoers` is out of scope for this fundamentals book.

```
student@linux:~$ apropos visudo
visudo               (8)  - edit the sudoers file
student@linux:~$
```

5.13. sudo su -

On some Linux systems like Ubuntu and Xubuntu, the `root` user does not have a password set. This means that it is not possible to login as `root` (extra security). To perform tasks as `root`, the first user is given all `sudo rights` via the `/etc/sudoers`. In fact all users that are members of the admin group can use sudo to run all commands as root.

```
root@linux:~# grep admin /etc/sudoers
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

The end result of this is that the user can type `sudo su -` and become root without having to enter the root password. The sudo command does require you to enter your own password. Thus the password prompt in the screenshot below is for sudo, not for su.

```
student@linux:~$ sudo su -
Password:
root@linux:~#
```

5.14. sudo logging

Using `sudo` without authorization will result in a severe warning:

```
student@linux:~$ sudo su -

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for paul:
paul is not in the sudoers file.  This incident will be reported.
student@linux:~$
```

The root user can see this in the `/var/log/secure` on Red Hat and in `/var/log/auth.log` on Debian).

```
root@linux:~# tail /var/log/secure | grep sudo | tr -s ' '
Apr 13 16:03:42 rhel65 sudo: paul : user NOT in sudoers ; TTY=pts/0 ; PWD=\
/home/paul ; USER=root ; COMMAND=/bin/su -
root@linux:~#
```

5.15. practice: introduction to users

1. Run a command that displays only your currently logged on user name.
2. Display a list of all logged on users.
3. Display a list of all logged on users including the command they are running at this very moment.
4. Display your user name and your unique user identification (userid).

5. Use `su` to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

6. Now use `su -` to switch to another user and notice the difference.

Note that `su -` gets you into the home directory of `Tania`.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

8. Now try the same, but with `sudo` before your command.

5.16. solution: introduction to users

1. Run a command that displays only your currently logged on user name.

```
laura@linux:~$ whoami
laura
laura@linux:~$ echo $USER
laura
```

2. Display a list of all logged on users.

```
laura@linux:~$ who
laura     pts/0        2014-10-13 07:22 (10.104.33.101)
laura@linux:~$
```

3. Display a list of all logged on users including the command they are running at this very moment.

```
laura@linux:~$ w
 07:47:02 up 16 min,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    10.104.33.101    07:30    6.00s  0.04s  0.00s w
root     pts/1    10.104.33.101    07:46    6.00s  0.01s  0.00s sleep 42
laura@linux:~$
```

4. Display your user name and your unique user identification (userid).

```
laura@linux:~$ id
uid=1005(laura) gid=1007(laura) groups=1007(laura)
laura@linux:~$
```

5. Use `su` to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

```
laura@linux:~$ su tania
Password:
tania@linux:/home/laura$ id
uid=1006(tania) gid=1008(tania) groups=1008(tania)
tania@linux:/home/laura$ exit
laura@linux:~$
```

6. Now use `su -` to switch to another user and notice the difference.

```
laura@linux:~$ su - tania
Password:
tania@linux:~$ pwd
/home/tania
tania@linux:~$ logout
laura@linux:~$
```

Note that `su -` gets you into the home directory of `Tania`.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

```
laura@linux:~$ useradd valentina
-su: useradd: command not found
laura@linux:~$ /usr/sbin/useradd valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

It is possible that `useradd` is located in `/sbin/useradd` on your computer.

8. Now try the same, but with `sudo` before your command.

```
laura@linux:~$ sudo /usr/sbin/useradd valentina
[sudo] password for laura:
laura is not in the sudoers file.  This incident will be reported.
laura@linux:~$
```

Notice that `laura` has no permission to use the `sudo` on this system.

6. user management

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This chapter will teach you how to use `useradd`, `usermod` and `userdel` to create, modify and remove user accounts.

You will need `root` access on a Linux computer to complete this chapter.

6.1. user management

User management on Linux can be done in three complementary ways. You can use the `graphical` tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice Linux user on your home system, then use the graphical tool that is provided by your distribution. This will make sure that you do not run into problems.

Another option is to use `command line tools` like useradd, usermod, gpasswd, passwd and others. Server administrators are likely to use these tools, since they are familiar and very similar across many different distributions. This chapter will focus on these command line tools.

A third and rather extremist way is to `edit the local configuration files` directly using vi (or vipw/vigr). Do not attempt this as a novice on production systems!

6.2. /etc/passwd

The local user database on Linux (and on most Unixes) is `/etc/passwd`.

```
[root@linux ~]# tail /etc/passwd
inge:x:518:524:art dealer:/home/inge:/bin/ksh
ann:x:519:525:flute player:/home/ann:/bin/bash
frederik:x:520:526:rubius poet:/home/frederik:/bin/bash
steven:x:521:527:roman emperor:/home/steven:/bin/bash
pascale:x:522:528:artist:/home/pascale:/bin/ksh
geert:x:524:530:kernel developer:/home/geert:/bin/bash
wim:x:525:531:master damuti:/home/wim:/bin/bash
sandra:x:526:532:radish stresser:/home/sandra:/bin/bash
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

As you can see, this file contains seven columns separated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

More information can be found by typing `man 5 passwd`.

```
[root@linux ~]# man 5 passwd
```

6.3. root

The `root` user also called the `superuser` is the most powerful account on your Linux system. This user can do almost anything, including the creation of other users. The root user always has userid 0 (regardless of the name of the account).

```
[root@linux ~]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

6.4. useradd

You can add users with the `useradd` command. The example below shows how to add a user named yanina (last parameter) and at the same time forcing the creation of the home directory (-m), setting the name of the home directory (-d), and setting a description (-c).

```
[root@linux ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@linux ~]# tail -1 /etc/passwd
yanina:x:529:529:yanina wickmayer:/home/yanina:/bin/bash
```

The user named yanina received userid 529 and `primary group` id 529.

6.5. /etc/default/useradd

Both Red Hat Enterprise Linux and Debian/Ubuntu have a file called `/etc/default/useradd` that contains some default user options. Besides using cat to display this file, you can also use `useradd -D`.

```
[root@RHEL4 ~]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

6.6. userdel

You can delete the user yanina with `userdel`. The -r option of userdel will also remove the home directory.

```
[root@linux ~]# userdel -r yanina
```

6.7. usermod

You can modify the properties of a user with the `usermod` command. This example uses `usermod` to change the description of the user harry.

```
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:harry potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

6.8. creating home directories

The easiest way to create a home directory is to supply the `-m` option with `useradd` (it is likely set as a default option on Linux).

A less easy way is to create a home directory manually with `mkdir` which also requires setting the owner and the permissions on the directory with `chmod` and `chown` (both commands are discussed in detail in another chapter).

```
[root@linux ~]# mkdir /home/laura
[root@linux ~]# chown laura:laura /home/laura
[root@linux ~]# chmod 700 /home/laura
[root@linux ~]# ls -ld /home/laura/
drwx------ 2 laura laura 4096 Jun 24 15:17 /home/laura/
```

6.9. /etc/skel/

When using `useradd` the `-m` option, the `/etc/skel/` directory is copied to the newly created home directory. The `/etc/skel/` directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way `/etc/skel/` serves as a default home directory and as a default user profile.

```
[root@linux ~]# ls -la /etc/skel/
total 48
drwxr-xr-x  2 root root  4096 Apr  1 00:11 .
drwxr-xr-x 97 root root 12288 Jun 24 15:36 ..
-rw-r--r--  1 root root    24 Jul 12  2006 .bash_logout
-rw-r--r--  1 root root   176 Jul 12  2006 .bash_profile
-rw-r--r--  1 root root   124 Jul 12  2006 .bashrc
```

6.10. deleting home directories

The -r option of `userdel` will make sure that the home directory is deleted together with the user account.

```
[root@linux ~]# ls -ld /home/wim/
drwx------ 2 wim wim 4096 Jun 24 15:19 /home/wim/
[root@linux ~]# userdel -r wim
[root@linux ~]# ls -ld /home/wim/
ls: /home/wim/: No such file or directory
```

6.11. login shell

The /etc/passwd file specifies the login shell for the user. In the screenshot below you can see that user annelies will log in with the /bin/bash shell, and user laura with the /bin/ksh shell.

```
[root@linux ~]# tail -2 /etc/passwd
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

You can use the usermod command to change the shell for a user.

```
[root@linux ~]# usermod -s /bin/bash laura
[root@linux ~]# tail -1 /etc/passwd
laura:x:528:534:art dealer:/home/laura:/bin/bash
```

6.12. chsh

Users can change their login shell with the chsh command. First, user harry obtains a list of available shells (he could also have done a cat /etc/shells) and then changes his login shell to the Korn shell (/bin/ksh). At the next login, harry will default into ksh instead of bash.

```
[laura@linux ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/ksh
/bin/tcsh
/bin/csh
[laura@linux ~]$
```

Note that the -l option does not exist on Debian and that the above screenshot assumes that ksh and csh shells are installed.

The screenshot below shows how laura can change her default shell (active on next login).

```
[laura@linux ~]$ chsh -s /bin/ksh
Changing shell for laura.
Password:
Shell changed.
```

6.13. practice: user management

1. Create a user account named serena, including a home directory and a description (or comment) that reads Serena Williams. Do all this in one single command.

2. Create a user named venus, including home directory, bash shell, a description that reads Venus Williams all in one single command.

3. Verify that both users have correct entries in /etc/passwd, /etc/shadow and /etc/group.

4. Verify that their home directory was created.

5. Create a user named `einstime` with `/bin/date` as his default logon shell.

6. What happens when you log on with the `einstime` user ? Can you think of a useful real world example for changing a user's login shell to an application ?

7. Create a file named `welcome.txt` and make sure every new user will see this file in their home directory.

8. Verify this setup by creating (and deleting) a test user account.

9. Change the default login shell for the `serena` user to `/bin/bash`. Verify before and after you make this change.

6.14. solution: user management

1. Create a user account named `serena`, including a home directory and a description (or comment) that reads `Serena Williams`. Do all this in one single command.

```
root@linux:~# useradd -m -c 'Serena Williams' serena
```

2. Create a user named `venus`, including home directory, bash shell, a description that reads `Venus Williams` all in one single command.

```
root@linux:~# useradd -m -c "Venus Williams" -s /bin/bash venus
```

3. Verify that both users have correct entries in `/etc/passwd`, `/etc/shadow` and `/etc/group`.

```
root@linux:~# tail -2 /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
venus:x:1009:1011:Venus Williams:/home/venus:/bin/bash
root@linux:~# tail -2 /etc/shadow
serena:!:16358:0:99999:7:::
venus:!:16358:0:99999:7:::
root@linux:~# tail -2 /etc/group
serena:x:1010:
venus:x:1011:
```

4. Verify that their home directory was created.

```
root@linux:~# ls -lrt /home | tail -2
drwxr-xr-x 2 serena    serena    4096 Oct 15 10:50 serena
drwxr-xr-x 2 venus     venus     4096 Oct 15 10:59 venus
root@linux:~#
```

5. Create a user named `einstime` with `/bin/date` as his default logon shell.

```
root@linux:~# useradd -s /bin/date einstime
```

Or even better:

```
root@linux:~# useradd -s $(which date) einstime
```

6. What happens when you log on with the `einstime` user ? Can you think of a useful real world example for changing a user's login shell to an application ?

```
root@linux:~# su - einstime
Wed Oct 15 11:05:56 UTC 2014    # You get the output of the date command
root@linux:~#
```

It can be useful when users need to access only one application on the server. Just logging in opens the application for them, and closing the application automatically logs them out.

7. Create a file named `welcome.txt` and make sure every new user will see this file in their home directory.

```
root@linux:~# echo Hello > /etc/skel/welcome.txt
```

8. Verify this setup by creating (and deleting) a test user account.

```
root@linux:~# useradd -m test
root@linux:~# ls -l /home/test
total 4
-rw-r--r-- 1 test test 6 Oct 15 11:16 welcome.txt
root@linux:~# userdel -r test
root@linux:~#
```

9. Change the default login shell for the `serena` user to `/bin/bash`. Verify before and after you make this change.

```
root@linux:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
root@linux:~# usermod -s /bin/bash serena
root@linux:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/bash
root@linux:~#
```

7. user passwords

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This chapter will tell you more about passwords for local users.

Three methods for setting passwords are explained; using the `passwd` command, using `openssel passwd`, and using the `crypt` function in a C program.

The chapter will also discuss password settings and disabling, suspending or locking accounts.

7.1. passwd

Passwords of users can be set with the `passwd` command. Users will have to provide their old password before twice entering the new one.

```
[tania@linux ~]$ passwd
Changing password for user tania.
Changing password for tania.
(current) UNIX password:
New password:
BAD PASSWORD: The password is shorter than 8 characters
New password:
BAD PASSWORD: The password is a palindrome
New password:
BAD PASSWORD: The password is too similar to the old one
passwd: Have exhausted maximum number of retries for service
```

As you can see, the passwd tool will do some basic verification to prevent users from using too simple passwords. The `root` user does not have to follow these rules (there will be a warning though). The `root` user also does not have to provide the old password before entering the new password twice.

```
root@linux:~# passwd tania
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

7.2. shadow file

User passwords are encrypted and kept in `/etc/shadow`. The /etc/shadow file is read only and can only be read by root. We will see in the file permissions section how it is possible for users to change their password. For now, you will have to know that users can change their password with the `/usr/bin/passwd` command.

```
[root@linux ~]# tail -4 /etc/shadow
paul:$6$ikp2Xta5BT.Tml.p$2TZjNnOYNNQKpwLJqoGJbVsZG5/Fti8ovBRd.VzRbiDSl7TEq\
IaSMH.TeBKnTS/SjlMruW8qffC0JNORW.BTW1:16338:0:99999:7:::
tania:$6$8Z/zovxj$9qvoqT8i9KIrmN.k4EQwAF5ryz5yzNwEvYjAa9L5XVXQu.z4DlpvMREH\
eQpQzvRnqFdKkVj17H5ST.c79HDZw0:16356:0:99999:7:::
laura:$6$glDuTY5e$/NYYWLxfHgZFWeoujaXSMcR.Mz.lGOxtcxFocFVJNb98nbTPhWFXfKWG\
SyYh1WCv6763Wq54.w24Yr3uAZBOm/:16356:0:99999:7:::
valentina:$6$jrZa6PVI$1uQgqR6En9mZB6mKJ3LXRB4CnFko6LRhbh.v4iqUk9MVreui1lv7\
GxHOUDSKA0N55ZRNhGHa6T2ouFnVno/0o1:16356:0:99999:7:::
[root@linux ~]#
```

The `/etc/shadow` file contains nine colon separated columns. The nine fields contain (from left to right) the user name, the encrypted password (note that only inge and laura have an encrypted password), the day the password was last changed (day 1 is January 1, 1970), number of days the password must be left unchanged, password expiry day, warning number of days before password expiry, number of days after expiry before disabling the account, and the day the account was disabled (again, since 1970). The last field has no meaning yet.

All the passwords in the screenshot above are hashes of `hunter2`.

7.3. encryption with passwd

Passwords are stored in an encrypted format. This encryption is done by the `crypt` function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the `useradd -m user` command, and then set the user's password with `passwd`.

```
[root@RHEL4 ~]# useradd -m xavier
[root@RHEL4 ~]# passwd xavier
Changing password for user xavier.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@RHEL4 ~]#
```

7.4. encryption with openssl

Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the `openssl passwd` command.

The `openssl passwd` command will generate several distinct hashes for the same password, for this it uses a `salt`.

```
student@linux:~$ openssl passwd hunter2
86jcUNlnGDFpY
student@linux:~$ openssl passwd hunter2
Yj7mDO9OAnvq6
student@linux:~$ openssl passwd hunter2
YqDcJeGoDbzKA
student@linux:~$
```

This `salt` can be chosen and is visible as the first two characters of the hash.

```
student@linux:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
student@linux:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
student@linux:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
student@linux:~$
```

This example shows how to create a user with password.

```
root@linux:~# useradd -m -p $(openssl passwd hunter2) mohamed
```

Note that this command puts the password in your command history!

7.5. encryption with crypt

A third option is to create your own C program using the crypt function, and compile this into a command.

```
student@linux:~$ cat MyCrypt.c
#include <stdio.h>
#define __USE_XOPEN
#include <unistd.h>

int main(int argc, char** argv)
{
 if(argc==3)
   {
       printf("%s\n", crypt(argv[1],argv[2]));
   }
   else
   {
       printf("Usage: MyCrypt $password $salt\n" );
   }
  return 0;
}
```

This little program can be compiled with `gcc` like this.

```
student@linux:~$ gcc MyCrypt.c -o MyCrypt -lcrypt
```

To use it, we need to give two parameters to MyCrypt. The first is the unencrypted password, the second is the salt. The salt is used to perturb the encryption algorithm in one of 4096 different ways. This variation prevents two users with the same password from having the same entry in `/etc/shadow`.

```
student@linux:~$ ./MyCrypt hunter2 42
42ZrbtP1Ze8G.
student@linux:~$ ./MyCrypt hunter2 33
33d6taYSiEUXI
```

Did you notice that the first two characters of the password are the `salt`?

The standard output of the crypt function is using the DES algorithm which is old and can be cracked in minutes. A better method is to use `md5` passwords which can be recognized by a salt starting with 1.

```
student@linux:~$ ./MyCrypt hunter2 '$1$42'
$1$42$7l6Y3xT5282XmZrtDOF9f0
student@linux:~$ ./MyCrypt hunter2 '$6$42'
$6$42$OqFFAVnI3gTSYG0yI9TZWX9cpyQzwIop7HwpG1LLEsNBiMr4w6OvLX1KDa./UpwXfrFk1i...
```

The `md5` salt can be up to eight characters long. The salt is displayed in `/etc/shadow` between the second and third $, so never use the password as the salt!

```
student@linux:~$ ./MyCrypt hunter2 '$1$hunter2'
$1$hunter2$YVxrxDmidq7Xf8Gdt6qM2.
```

7.6. /etc/login.defs

The `/etc/login.defs` file contains some default settings for user passwords like password aging and length settings. (You will also find the numerical limits of user ids and group ids and whether or not a home directory should be created by default).

```
root@linux:~# grep ^PASS /etc/login.defs
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

Debian also has this file.

```
root@linux:~# grep PASS /etc/login.defs
#  PASS_MAX_DAYS   Maximum number of days a password may be used.
#  PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#  PASS_WARN_AGE   Number of days warning given before a password expires.
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
#PASS_CHANGE_TRIES
#PASS_ALWAYS_WARN
#PASS_MIN_LEN
#PASS_MAX_LEN
# NO_PASSWORD_CONSOLE
root@linux:~#
```

7.7. chage

The `chage` command can be used to set an expiration date for a user account (-E), set a minimum (-m) and maximum (-M) password age, a password expiration date, and set the number of warning days before the password expiration date. Much of this functionality is also available from the `passwd` command. The `-l` option of chage will list these settings for a user.

```
root@linux:~# chage -l paul
Last password change                                    : Mar 27, 2014
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
```

```
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
root@linux:~#
```

7.8. disabling a password

Passwords in `/etc/shadow` cannot begin with an exclamation mark. When the second field in `/etc/passwd` starts with an exclamation mark, then the password can not be used.

Using this feature is often called `locking`, `disabling`, or `suspending` a user account. Besides `vi` (or vipw) you can also accomplish this with `usermod`.

The first command in the next screenshot will show the hashed password of `laura` in `/etc/shadow`. The next command disables the password of `laura`, making it impossible for Laura to authenticate using this password.

```
root@linux:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
root@linux:~# usermod -L laura
```

As you can see below, the password hash is simply preceded with an exclamation mark.

```
root@linux:~# grep laura /etc/shadow | cut -c1-70
laura:!$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJ
root@linux:~#
```

The root user (and users with `sudo` rights on `su`) still will be able to `su` into the `laura` account (because the password is not needed here). Also note that `laura` will still be able to login if she has set up passwordless ssh!

```
root@linux:~# su - laura
laura@linux:~$
```

You can unlock the account again with `usermod -U`.

```
root@linux:~# usermod -U laura
root@linux:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
```

Watch out for tiny differences in the command line options of `passwd`, `usermod`, and `useradd` on different Linux distributions. Verify the local files when using features like `"disabling, suspending, or locking"` on user accounts and their passwords.

7.9. editing local files

If you still want to manually edit the `/etc/passwd` or `/etc/shadow`, after knowing these commands for password management, then use `vipw` instead of vi(m) directly. The `vipw` tool will do proper locking of the file.

```
[root@linux ~]# vipw /etc/passwd
vipw: the password file is busy (/etc/ptmp present)
```

7.10. practice: user passwords

1. Set the password for `serena` to `hunter2`.

2. Also set a password for `venus` and then lock the `venus` user account with `usermod`. Verify the locking in `/etc/shadow` before and after you lock it.

3. Use `passwd -d` to disable the `serena` password. Verify the `serena` line in `/etc/shadow` before and after disabling.

4. What is the difference between locking a user account and disabling a user account's password like we just did with `usermod -L` and `passwd -d`?

5. Try changing the password of serena to serena as serena.

6. Make sure `serena` has to change her password in 10 days.

7. Make sure every new user needs to change their password every 10 days.

8. Take a backup as root of `/etc/shadow`. Use `vi` to copy an encrypted `hunter2` hash from `venus` to `serena`. Can `serena` now log on with `hunter2` as a password ?

9. Why use `vipw` instead of `vi` ? What could be the problem when using `vi` or `vim` ?

10. Use `chsh` to list all shells (only works on RHEL/CentOS/Fedora), and compare to `cat /etc/shells`.

11. Which `useradd` option allows you to name a home directory ?

12. How can you see whether the password of user `serena` is locked or unlocked ? Give a solution with `grep` and a solution with `passwd`.

7.11. solution: user passwords

1. Set the password for `serena` to `hunter2`.

```
root@linux:~# passwd serena
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2. Also set a password for `venus` and then lock the `venus` user account with `usermod`. Verify the locking in `/etc/shadow` before and after you lock it.

```
root@linux:~# passwd venus
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@linux:~# grep venus /etc/shadow | cut -c1-70
venus:$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM/
root@linux:~# usermod -L venus
root@linux:~# grep venus /etc/shadow | cut -c1-70
venus:!$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM
```

Note that `usermod -L` precedes the password hash with an exclamation mark (!).

3. Use `passwd -d` to disable the `serena` password. Verify the `serena` line in `/etc/shadow` before and after disabling.

```
root@linux:~# grep serena /etc/shadow | cut -c1-70
serena:$6$Es/omrPE$F2Ypu8kpLrfKdW0v/UIwA5jrYyBD2nwZ/dt.i/IypRgiPZSdB/B
root@linux:~# passwd -d serena
passwd: password expiry information changed.
root@linux:~# grep serena /etc/shadow
serena::16358:0:99999:7:::
root@linux:~#
```

4. What is the difference between locking a user account and disabling a user account's password like we just did with `usermod -L` and `passwd -d`?

Locking will prevent the user from logging on to the system with his password by putting a ! in front of the password in `/etc/shadow`.

Disabling with `passwd` will erase the password from `/etc/shadow`.

5. Try changing the password of serena to serena as serena.

```
log on as serena, then execute: passwd serena ... it should fail!
```

6. Make sure `serena` has to change her password in 10 days.

```
chage -M 10 serena
```

7. Make sure every new user needs to change their password every 10 days.

```
vi /etc/login.defs (and change PASS_MAX_DAYS to 10)
```

8. Take a backup as root of `/etc/shadow`. Use `vi` to copy an encrypted `hunter2` hash from `venus` to `serena`. Can `serena` now log on with `hunter2` as a password ?

```
Yes.
```

9. Why use `vipw` instead of `vi` ? What could be the problem when using `vi` or `vim` ?

```
vipw will give a warning when someone else is already using that file (with vipw).
```

10. Use `chsh` to list all shells (only works on RHEL/CentOS/Fedora), and compare to `cat /etc/shells`.

```
chsh -l
cat /etc/shells
```

11. Which `useradd` option allows you to name a home directory ?

```
-d
```

12. How can you see whether the password of user `serena` is locked or unlocked ? Give a solution with `grep` and a solution with `passwd`.

```
grep serena /etc/shadow
```

```
passwd -S serena
```

8. User profiles

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

Logged on users have a number of preset (and customized) aliases, variables, and functions, but where do they come from ? The `shell` uses a number of startup files that are executed (or rather `sourced`) whenever the shell is invoked. What follows is an overview of startup scripts.

8.1. system profile

Both the `bash` and the `ksh` shell will verify the existence of `/etc/profile` and `source` it if it exists.

When reading this script, you will notice (both on Debian and on Red Hat Enterprise Linux) that it builds the PATH environment variable (among others). The script might also change the PS1 variable, set the HOSTNAME and execute even more scripts like `/etc/inputrc`

This screenshot uses grep to show PATH manipulation in `/etc/profile` on Debian.

```
root@linux:~# grep PATH /etc/profile
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
export PATH
root@linux:~#
```

This screenshot uses grep to show PATH manipulation in `/etc/profile` on RHEL7/CentOS7.

```
[root@linux ~]# grep PATH /etc/profile
    case ":${PATH}:" in
                PATH=$PATH:$1
                PATH=$1:$PATH
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
[root@linux ~]#
```

The `root user` can use this script to set aliases, functions, and variables for every user on the system.

8.2. ~/.bash_profile

When this file exists in the home directory, then `bash` will source it. On Debian Linux 5/6/7 this file does not exist by default.

RHEL7/CentOS7 uses a small `~/.bash_profile` where it checks for the existence of `~/.bashrc` and then sources it. It also adds $HOME/bin to the $PATH variable.

```
[root@linux ~]# cat /home/paul/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
[root@linux ~]#
```

8.3. ~/.bash_login

When `.bash_profile` does not exist, then `bash` will check for `~/.bash_login` and source it.

Neither Debian nor Red Hat have this file by default.

8.4. ~/.profile

When neither `~/.bash_profile` and `~/.bash_login` exist, then bash will verify the existence of `~/.profile` and execute it. This file does not exist by default on Red Hat.

On Debian this script can execute `~/.bashrc` and will add $HOME/bin to the $PATH variable.

```
root@linux:~# tail -11 /home/paul/.profile
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

RHEL/CentOS does not have this file by default.

8.5. ~/.bashrc

The `~/.bashrc` script is often sourced by other scripts. Let us take a look at what it does by default.

Red Hat uses a very simple `~/.bashrc`, checking for `/etc/bashrc` and sourcing it. It also leaves room for custom aliases and functions.

```
[root@linux ~]# cat /home/paul/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-
paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

On Debian this script is quite a bit longer and configures $PS1, some history variables and a number af active and inactive aliases.

```
root@linux:~# wc -l /home/paul/.bashrc
110 /home/paul/.bashrc
```

8.6. ~/.bash_logout

When exiting `bash`, it can execute `~/.bash_logout`.

Debian use this opportunity to clear the console screen.

```
serena@linux:~$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```

Red Hat Enterprise Linux 5 will simple call the `/usr/bin/clear` command in this script.

```
[serena@linux ~]$ cat .bash_logout
# ~/.bash_logout

/usr/bin/clear
```

Red Hat Enterprise Linux 6 and 7 create this file, but leave it empty (except for a comment).

```
student@linux:~$ cat .bash_logout
# ~/.bash_logout
```

8.7. Debian overview

Below is a table overview of when Debian is running any of these bash startup scripts.

Table 8.1.: Debian User Environment

script	su	su -	ssh	gdm
~./bashrc	no	yes	yes	yes
~/.profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bash.bashrc	yes	no	no	yes

8.8. RHEL5 overview

Below is a table overview of when Red Hat Enterprise Linux 5 is running any of these bash startup scripts.

Table 8.2.: Red Hat User Environment

script	su	su -	ssh	gdm
~./bashrc	yes	yes	yes	yes
~/.bash_profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bashrc	yes	yes	yes	yes

8.9. practice: user profiles

1. Make a list of all the profile files on your system.

2. Read the contents of each of these, often they `source` extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

8.10. solution: user profiles

1. Make a list of all the profile files on your system.

```
ls -a ~ ; ls -l /etc/pro* /etc/bash*
```

2. Read the contents of each of these, often they `source` extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

```
same name aliases, functions and variables will overwrite each other
```

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

```
Yes it does matter. (man bash /INVOCATION)
```

9. groups

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

Users can be listed in `groups`. Groups allow you to set permissions on the group level instead of having to set permissions for every individual user.

Every Unix or Linux distribution will have a graphical tool to manage groups. Novice users are advised to use this graphical tool. More experienced users can use command line tools to manage users, but be careful: Some distributions do not allow the mixed use of GUI and CLI tools to manage groups (YaST in Novell Suse). Senior administrators can edit the relevant files directly with `vi` or `vigr`.

9.1. groupadd

Groups can be created with the `groupadd` command. The example below shows the creation of five (empty) groups.

```
root@linux:~# groupadd tennis
root@linux:~# groupadd football
root@linux:~# groupadd snooker
root@linux:~# groupadd formula1
root@linux:~# groupadd salsa
```

9.2. group file

Users can be a member of several groups. Group membership is defined by the `/etc/group` file.

```
root@linux:~# tail -5 /etc/group
tennis:x:1006:
football:x:1007:
snooker:x:1008:
formula1:x:1009:
salsa:x:1010:
root@linux:~#
```

The first field is the group's name. The second field is the group's (encrypted) password (can be empty). The third field is the group identification or `GID`. The fourth field is the list of members, these groups have no members.

9.3. groups

A user can type the `groups` command to see a list of groups where the user belongs to.

```
[harry@linux ~]$ groups
harry sports
[harry@linux ~]$
```

9.4. usermod

Group membership can be modified with the useradd or `usermod` command.

```
root@linux:~# usermod -a -G tennis inge
root@linux:~# usermod -a -G tennis katrien
root@linux:~# usermod -a -G salsa katrien
root@linux:~# usermod -a -G snooker sandra
root@linux:~# usermod -a -G formula1 annelies
root@linux:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
snooker:x:1008:sandra
formula1:x:1009:annelies
salsa:x:1010:katrien
root@linux:~#
```

Be careful when using `usermod` to add users to groups. By default, the `usermod` command will `remove` the user from every group of which he is a member if the group is not listed in the command! Using the `-a` (append) switch prevents this behaviour.

9.5. groupmod

You can change the group name with the `groupmod` command.

```
root@linux:~# groupmod -n darts snooker
root@linux:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
formula1:x:1009:annelies
salsa:x:1010:katrien
darts:x:1008:sandra
```

9.6. groupdel

You can permanently remove a group with the `groupdel` command.

```
root@linux:~# groupdel tennis
root@linux:~#
```

9.7. gpasswd

You can delegate control of group membership to another user with the `gpasswd` command. In the example below we delegate permissions to add and remove group members to serena for the sports group. Then we `su` to serena and add harry to the sports group.

```
[root@linux ~]# gpasswd -A serena sports
[root@linux ~]# su - serena
[serena@linux ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry)
[serena@linux ~]$ gpasswd -a harry sports
Adding user harry to group sports
[serena@linux ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry),522(sports)
[serena@linux ~]$ tail -1 /etc/group
sports:x:522:serena,venus,harry
[serena@linux ~]$
```

Group administrators do not have to be a member of the group. They can remove themselves from a group, but this does not influence their ability to add or remove members.

```
[serena@linux ~]$ gpasswd -d serena sports
Removing user serena from group sports
[serena@linux ~]$ exit
```

Information about group administrators is kept in the `/etc/gshadow` file.

```
[root@linux ~]# tail -1 /etc/gshadow
sports:!:serena:venus,harry
[root@linux ~]#
```

To remove all group administrators from a group, use the `gpasswd` command to set an empty administrators list.

```
[root@linux ~]# gpasswd -A "" sports
```

9.8. newgrp

You can start a `child shell` with a new temporary `primary group` using the `newgrp` command.

```
root@linux:~# mkdir prigroup
root@linux:~# cd prigroup/
root@linux:~/prigroup# touch standard.txt
root@linux:~/prigroup# ls -l
total 0
-rw-r--r--. 1 root root 0 Apr 13 17:49 standard.txt
root@linux:~/prigroup# echo $SHLVL
1
root@linux:~/prigroup# newgrp tennis
root@linux:~/prigroup# echo $SHLVL
2
root@linux:~/prigroup# touch newgrp.txt
root@linux:~/prigroup# ls -l
```

```
total 0
-rw-r--r--. 1 root tennis 0 Apr 13 17:49 newgrp.txt
-rw-r--r--. 1 root root   0 Apr 13 17:49 standard.txt
root@linux:~/prigroup# exit
exit
root@linux:~/prigroup#
```

9.9. vigr

Similar to vipw, the `vigr` command can be used to manually edit the `/etc/group` file, since it will do proper locking of the file. Only experienced senior administrators should use `vi` or `vigr` to manage groups.

9.10. practice: groups

1. Create the groups tennis, football and sports.
2. In one command, make venus a member of tennis and sports.
3. Rename the football group to foot.
4. Use vi to add serena to the tennis group.
5. Use the id command to verify that serena is a member of tennis.
6. Make someone responsible for managing group membership of foot and sports. Test that it works.

9.11. solution: groups

1. Create the groups tennis, football and sports.

```
groupadd tennis ; groupadd football ; groupadd sports
```

2. In one command, make venus a member of tennis and sports.

```
usermod -a -G tennis,sports venus
```

3. Rename the football group to foot.

```
groupmod -n foot football
```

4. Use vi to add serena to the tennis group.

```
vi /etc/group
```

5. Use the id command to verify that serena is a member of tennis.

```
id (and after logoff logon serena should be member)
```

6. Make someone responsible for managing group membership of foot and sports. Test that it works.

```
gpasswd -A (to make manager)

gpasswd -a (to add member)
```

Part III.

Scripting 101

10. I/O redirection

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

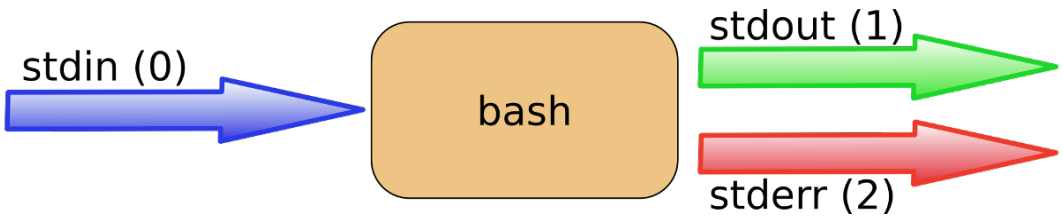
One of the powers of the Unix command line is the use of `input/output redirection` and `pipes`.

This chapter explains `redirection` of input, output and error streams.

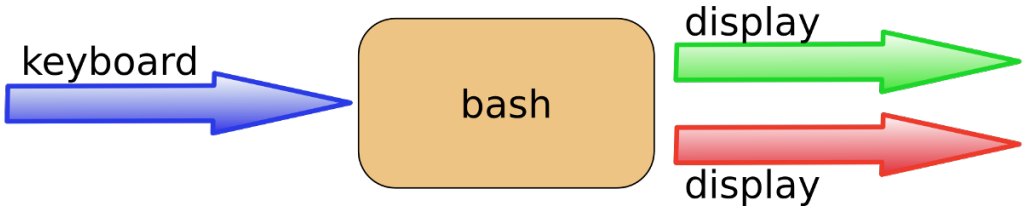
10.1. stdin, stdout, and stderr

The bash shell has three basic streams; it takes input from `stdin` (stream `0`), it sends output to `stdout` (stream 1) and it sends error messages to `stderr` (stream 2) .

The drawing below has a graphical interpretation of these three streams.



The keyboard often serves as `stdin`, whereas `stdout` and `stderr` both go to the display. This can be confusing to new Linux users because there is no obvious way to recognize `stdout` from `stderr`. Experienced users know that separating output from errors can be very useful.

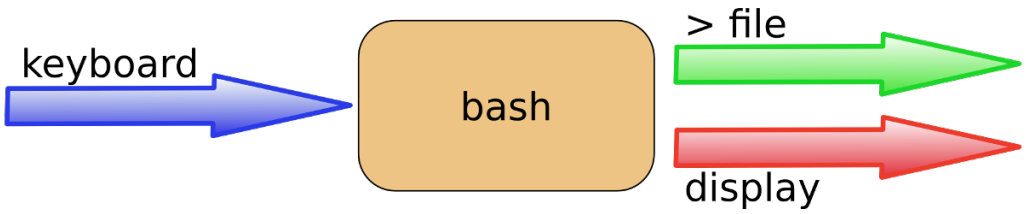


The next sections will explain how to redirect these streams.

10.2. output redirection

10.2.1. > stdout

`stdout` can be redirected with a `greater than` sign. While scanning the line, the shell will see the > sign and will clear the file.



The > notation is in fact the abbreviation of 1> (`stdout` being referred to as stream 1).

```
[student@linux ~]$ echo It is cold today!
It is cold today!
[student@linux ~]$ echo It is cold today! > winter.txt
[student@linux ~]$ cat winter.txt
It is cold today!
[student@linux ~]$
```

Note that the bash shell effectively `removes` the redirection from the command line before argument 0 is executed. This means that in the case of this command:

```
echo hello > greetings.txt
```

the shell only counts two arguments (echo = argument 0, hello = argument 1). The redirection is removed before the argument counting takes place.

10.2.2. output file is erased

While scanning the line, the shell will see the > sign and `will clear the file`! Since this happens before resolving `argument 0`, this means that even when the command fails, the file will have been cleared!

```
[student@linux ~]$ cat winter.txt
It is cold today!
[student@linux ~]$ zcho It is cold today! > winter.txt
-bash: zcho: command not found
[student@linux ~]$ cat winter.txt
[student@linux ~]$
```

10.2.3. noclobber

Erasing a file while using > can be prevented by setting the `noclobber` option.

```
[student@linux ~]$ cat winter.txt
It is cold today!
[student@linux ~]$ set -o noclobber
[student@linux ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[student@linux ~]$ set +o noclobber
[student@linux ~]$
```

10.2.4. overruling noclobber

The `noclobber` can be overruled with `>|`.

```
[student@linux ~]$ set -o noclobber
[student@linux ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[student@linux ~]$ echo It is very cold today! >| winter.txt
[student@linux ~]$ cat winter.txt
It is very cold today!
[student@linux ~]$
```

10.2.5. » append

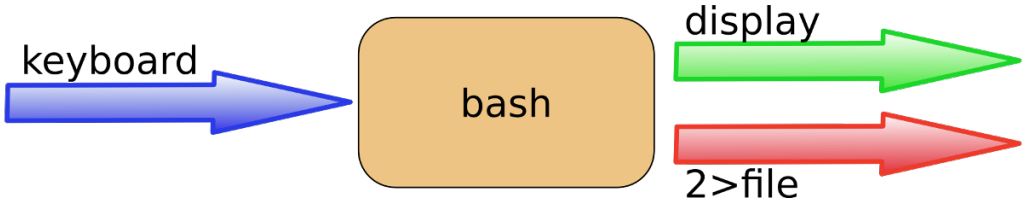
Use `>>` to `append` output to a file.

```
[student@linux ~]$ echo It is cold today! > winter.txt
[student@linux ~]$ cat winter.txt
It is cold today!
[student@linux ~]$ echo Where is the summer ? >> winter.txt
[student@linux ~]$ cat winter.txt
It is cold today!
Where is the summer ?
[student@linux ~]$
```

10.3. error redirection

10.3.1. 2> stderr

Redirecting `stderr` is done with `2>`. This can be very useful to prevent error messages from cluttering your screen.



The screenshot below shows redirection of `stdout` to a file, and `stderr` to `/dev/null`. Writing `1>` is the same as `>`.

```
[student@linux ~]$ find / > allfiles.txt 2> /dev/null
[student@linux ~]$
```

10.3.2. 2>&1

To redirect both `stdout` and `stderr` to the same file, use `2>&1`.

```
[student@linux ~]$ find / > allfiles_and_errors.txt 2>&1
[student@linux ~]$
```

Note that the order of redirections is significant. For example, the command

```
ls > dirlist 2>&1
```

directs both standard output (file descriptor 1) and standard error (file descriptor 2) to the file dirlist, while the command

```
ls 2>&1 > dirlist
```

directs only the standard output to file dirlist, because the standard error made a copy of the standard output before the standard output was redirected to dirlist.

10.4. output redirection and pipes

By default you cannot grep inside `stderr` when using pipes on the command line, because only `stdout` is passed.

```
student@linux:~$ rm file42 file33 file1201 | grep file42
rm: cannot remove ‘file42’: No such file or directory
rm: cannot remove ‘file33’: No such file or directory
rm: cannot remove ‘file1201’: No such file or directory
```

With `2>&1` you can force `stderr` to go to `stdout`. This enables the next command in the pipe to act on both streams.

```
student@linux:~$ rm file42 file33 file1201 2>&1 | grep file42
rm: cannot remove ‘file42’: No such file or directory
```

You cannot use both `1>&2` and `2>&1` to switch `stdout` and `stderr`.

```
student@linux:~$ rm file42 file33 file1201 2>&1 1>&2 | grep file42
rm: cannot remove ‘file42’: No such file or directory
student@linux:~$ echo file42 2>&1 1>&2 | sed 's/file42/FILE42/'
FILE42
```

You need a third stream to switch stdout and stderr after a pipe symbol.

```
student@linux:~$ echo file42 3>&1 1>&2 2>&3 | sed 's/file42/FILE42/'
file42
student@linux:~$ rm file42 3>&1 1>&2 2>&3 | sed 's/file42/FILE42/'
rm: cannot remove ‘FILE42’: No such file or directory
```

10.5. joining stdout and stderr

The `&>` construction will put both `stdout` and `stderr` in one stream (to a file).

```
student@linux:~$ rm file42 &> out_and_err
student@linux:~$ cat out_and_err
rm: cannot remove ‘file42’: No such file or directory
student@linux:~$ echo file42 &> out_and_err
student@linux:~$ cat out_and_err
file42
student@linux:~$
```

10.6. input redirection

10.6.1. < stdin

Redirecting `stdin` is done with < (short for 0<).

```
[student@linux ~]$ cat < text.txt
one
two
[student@linux ~]$ tr 'onetw' 'ONEZZ' < text.txt
ONE
ZZO
[student@linux ~]$
```

10.6.2. « here document

The `here document` (sometimes called here-is-document) is a way to append input until a certain sequence (usually EOF) is encountered. The `EOF` marker can be typed literally or can be called with Ctrl-D.

```
[student@linux ~]$ cat <<EOF > text.txt
> one
> two
> EOF
[student@linux ~]$ cat text.txt
one
two
[student@linux ~]$ cat <<brol > text.txt
> brel
> brol
[student@linux ~]$ cat text.txt
brel
[student@linux ~]$
```

10.6.3. «< here string

The `here string` can be used to directly pass strings to a command. The result is the same as using `echo string | command` (but you have one less process running).

```
student@linux~$ base64 <<< linux-training.be
bGludXgtdHJhaW5pbmcuYmUK
student@linux~$ base64 -d <<< bGludXgtdHJhaW5pbmcuYmUK
linux-training.be
```

See rfc 3548 for more information about `base64`.

10.7. confusing redirection

The shell will scan the whole line before applying redirection. The following command line is very readable and is correct.

```
cat winter.txt > snow.txt 2> errors.txt
```

But this one is also correct, but less readable.

```
2> errors.txt cat winter.txt > snow.txt
```

Even this will be understood perfectly by the shell.

```
< winter.txt > snow.txt 2> errors.txt cat
```

10.8. quick file clear

So what is the quickest way to clear a file ?

```
>foo
```

And what is the quickest way to clear a file when the `noclobber` option is set ?

```
>|bar
```

10.9. practice: input/output redirection

1. Activate the `noclobber` shell option.

2. Verify that `noclobber` is active by repeating an `ls` on `/etc/` with redirected output to a file.

3. When listing all shell options, which character represents the `noclobber` option ?

4. Deactivate the `noclobber` option.

5. Make sure you have two shells open on the same computer. Create an empty `tailing.txt` file. Then type `tail -f tailing.txt`. Use the second shell to `append` a line of text to that file. Verify that the first shell displays this line.

6. Create a file that contains the names of five people. Use `cat` and output redirection to create the file and use a `here document` to end the input.

10.10. solution: input/output redirection

1. Activate the `noclobber` shell option.

```
set -o noclobber
set -C
```

2. Verify that `noclobber` is active by repeating an `ls` on `/etc/` with redirected output to a file.

```
ls /etc > etc.txt
ls /etc > etc.txt (should not work)
```

3. When listing all shell options, which character represents the `noclobber` option ?

```
echo $- (noclobber is visible as C)
```

4. Deactivate the `noclobber` option.

```
set +o noclobber
```

5. Make sure you have two shells open on the same computer. Create an empty `tailing.txt` file. Then type `tail -f tailing.txt`. Use the second shell to `append` a line of text to that file. Verify that the first shell displays this line.

```
student@linux:~$ > tailing.txt
student@linux:~$ tail -f tailing.txt
hello
world

in the other shell:
student@linux:~$ echo hello >> tailing.txt
student@linux:~$ echo world >> tailing.txt
```

6. Create a file that contains the names of five people. Use `cat` and output redirection to create the file and use a `here document` to end the input.

```
student@linux:~$ cat > tennis.txt << ace
> Justine Henin
> Venus Williams
> Serena Williams
> Martina Hingis
> Kim Clijsters
> ace
student@linux:~$ cat tennis.txt
Justine Henin
Venus Williams
Serena Williams
Martina Hingis
Kim Clijsters
student@linux:~$
```

11. filters

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

Commands that are created to be used with a `pipe` are often called `filters`. These `filters` are very small programs that do one specific thing very efficiently. They can be used as `building blocks`.

This chapter will introduce you to the most common `filters`. The combination of simple commands and filters in a long `pipe` allows you to design elegant solutions.

11.1. cat

When between two `pipes`, the `cat` command does nothing (except putting `stdin` on `stdout`).

```
[student@linux pipes]$ tac count.txt | cat | cat | cat | cat | cat
five
four
three
two
one
[student@linux pipes]$
```

11.2. tee

Writing long `pipes` in Unix is fun, but sometimes you may want intermediate results. This is were `tee` comes in handy. The `tee` filter puts `stdin` on `stdout` and also into a file. So `tee` is almost the same as `cat`, except that it has two identical outputs.

```
[student@linux pipes]$ tac count.txt | tee temp.txt | tac
one
two
three
four
five
[student@linux pipes]$ cat temp.txt
five
four
three
two
one
[student@linux pipes]$
```

11.3. grep

The `grep` filter is famous among Unix users. The most common use of `grep` is to filter lines of text containing (or not containing) a certain string.

```
[student@linux pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[student@linux pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
Venus Williams, USA
```

You can write this without the cat.

```
[student@linux pipes]$ grep Williams tennis.txt
Serena Williams, usa
Venus Williams, USA
```

One of the most useful options of grep is `grep -i` which filters in a case insensitive way.

```
[student@linux pipes]$ grep Bel tennis.txt
Justine Henin, Bel
[student@linux pipes]$ grep -i Bel tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
[student@linux pipes]$
```

Another very useful option is `grep -v` which outputs lines not matching the string.

```
[student@linux pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[student@linux pipes]$
```

And of course, both options can be combined to filter all lines not containing a case insensitive string.

```
[student@linux pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
[student@linux pipes]$
```

With `grep -A1` one line `after` the result is also displayed.

```
student@linux:~/pipes$ grep -A1 Henin tennis.txt
Justine Henin, Bel
Serena Williams, usa
```

With `grep -B1` one line `before` the result is also displayed.

```
student@linux:~/pipes$ grep -B1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
```

With `grep -C1` (context) one line `before` and one `after` are also displayed. All three options (A,B, and C) can display any number of lines (using e.g. A2, B4 or C20).

```
student@linux:~/pipes$ grep -C1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
```

11.4. cut

The `cut` filter can select columns from files, depending on a delimiter or a count of bytes. The screenshot below uses `cut` to filter for the username and userid in the `/etc/passwd` file. It uses the colon as a delimiter, and selects fields 1 and 3.

```
[[student@linux pipes]$ cut -d: -f1,3 /etc/passwd | tail -4
Figo:510
Pfaff:511
Harry:516
Hermione:517
[student@linux pipes]$
```

When using a space as the delimiter for `cut`, you have to quote the space.

```
[student@linux pipes]$ cut -d" " -f1 tennis.txt
Amelie
Kim
Justine
Serena
Venus
[student@linux pipes]$
```

This example uses `cut` to display the second to the seventh character of `/etc/passwd`.

```
[student@linux pipes]$ cut -c2-7 /etc/passwd | tail -4
igo:x:
faff:x
arry:x
ermion
[student@linux pipes]$
```

11.5. tr

You can translate characters with `tr`. The screenshot shows the translation of all occurrences of e to E.

```
[student@linux pipes]$ cat tennis.txt | tr 'e' 'E'
AmEliE MaurEsmo, Fra
Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
VEnus Williams, USA
```

Here we set all letters to uppercase by defining two ranges.

```
[student@linux pipes]$ cat tennis.txt | tr 'a-z' 'A-Z'
AMELIE MAURESMO, FRA
KIM CLIJSTERS, BEL
JUSTINE HENIN, BEL
SERENA WILLIAMS, USA
VENUS WILLIAMS, USA
[student@linux pipes]$
```

Here we translate all newlines to spaces.

```
[student@linux pipes]$ cat count.txt
one
two
three
four
five
[student@linux pipes]$ cat count.txt | tr '\n' ' '
one two three four five [student@linux pipes]$
```

The `tr -s` filter can also be used to squeeze multiple occurrences of a character to one.

```
[student@linux pipes]$ cat spaces.txt
one    two        three
     four   five  six
[student@linux pipes]$ cat spaces.txt | tr -s ' '
one two three
 four five six
[student@linux pipes]$
```

You can also use `tr` to 'encrypt' texts with `rot13`.

```
[student@linux pipes]$ cat count.txt | tr 'a-z' 'nopqrstuvwxyzabcdefghijklm'
bar
gjb
guerr
sbhe
svir
[student@linux pipes]$ cat count.txt | tr 'a-z' 'n-za-m'
bar
gjb
guerr
sbhe
svir
[student@linux pipes]$
```

This last example uses `tr -d` to delete characters.

```
student@linux:~/pipes$ cat tennis.txt | tr -d e
Amli Maursmo, Fra
Kim Clijstrs, BEL
Justin Hnin, Bl
Srna Williams, usa
Vnus Williams, USA
```

11.6. wc

Counting words, lines and characters is easy with `wc`.

```
[student@linux pipes]$ wc tennis.txt
  5  15 100 tennis.txt
[student@linux pipes]$ wc -l tennis.txt
5 tennis.txt
[student@linux pipes]$ wc -w tennis.txt
15 tennis.txt
[student@linux pipes]$ wc -c tennis.txt
100 tennis.txt
[student@linux pipes]$
```

11.7. sort

The `sort` filter will default to an alphabetical sort.

```
student@linux:~/pipes$ cat music.txt
Queen
Brel
Led Zeppelin
Abba
student@linux:~/pipes$ sort music.txt
Abba
Brel
Led Zeppelin
Queen
```

But the `sort` filter has many options to tweak its usage. This example shows sorting different columns (column 1 or column 2).

```
[student@linux pipes]$ sort -k1 country.txt
Belgium, Brussels, 10
France, Paris, 60
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50
[student@linux pipes]$ sort -k2 country.txt
Germany, Berlin, 100
Belgium, Brussels, 10
France, Paris, 60
Italy, Rome, 50
Iran, Teheran, 70
```

The screenshot below shows the difference between an alphabetical sort and a numerical sort (both on the third column).

```
[student@linux pipes]$ sort -k3 country.txt
Belgium, Brussels, 10
Germany, Berlin, 100
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
[student@linux pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
Germany, Berlin, 100
```

11.8. uniq

With `uniq` you can remove duplicates from a `sorted list`.

```
student@linux:~/pipes$ cat music.txt
Queen
Brel
Queen
Abba
student@linux:~/pipes$ sort music.txt
Abba
Brel
Queen
Queen
student@linux:~/pipes$ sort music.txt |uniq
Abba
Brel
Queen
```

`uniq` can also count occurrences with the `-c` option.

```
student@linux:~/pipes$ sort music.txt |uniq -c
      1 Abba
      1 Brel
      2 Queen
```

11.9. comm

Comparing streams (or files) can be done with the `comm`. By default `comm` will output three columns. In this example, Abba, Cure and Queen are in both lists, Bowie and Sweet are only in the first file, Turner is only in the second.

```
student@linux:~/pipes$ cat > list1.txt
Abba
Bowie
Cure
Queen
```

```
Sweet
student@linux:~/pipes$ cat > list2.txt
Abba
Cure
Queen
Turner
student@linux:~/pipes$ comm list1.txt list2.txt
                Abba
Bowie
                Cure
                Queen
Sweet
        Turner
```

The output of `comm` can be easier to read when outputting only a single column. The digits point out which output columns should not be displayed.

```
student@linux:~/pipes$ comm -12 list1.txt list2.txt
Abba
Cure
Queen
student@linux:~/pipes$ comm -13 list1.txt list2.txt
Turner
student@linux:~/pipes$ comm -23 list1.txt list2.txt
Bowie
Sweet
```

11.10. od

European humans like to work with ascii characters, but computers store files in bytes. The example below creates a simple file, and then uses `od` to show the contents of the file in hexadecimal bytes

```
student@linux:~/test$ cat > text.txt
abcdefg
1234567
student@linux:~/test$ od -t x1 text.txt
0000000 61 62 63 64 65 66 67 0a 31 32 33 34 35 36 37 0a
0000020
```

The same file can also be displayed in octal bytes.

```
student@linux:~/test$ od -b text.txt
0000000 141 142 143 144 145 146 147 012 061 062 063 064 065 066 067 012
0000020
```

And here is the file in ascii (or backslashed) characters.

```
student@linux:~/test$ od -c text.txt
0000000   a   b   c   d   e   f   g  \n   1   2   3   4   5   6   7  \n
0000020
```

11.11. sed

The stream editor `sed` can perform editing functions in the stream, using `regular expressions`.

```
student@linux:~/pipes$ echo level5 | sed 's/5/42/'
level42
student@linux:~/pipes$ echo level5 | sed 's/level/jump/'
jump5
```

Add `g` for global replacements (all occurrences of the string per line).

```
student@linux:~/pipes$ echo level5 level7 | sed 's/level/jump/'
jump5 level7
student@linux:~/pipes$ echo level5 level7 | sed 's/level/jump/g'
jump5 jump7
```

With `d` you can remove lines from a stream containing a character.

```
student@linux:~/test42$ cat tennis.txt
Venus Williams, USA
Martina Hingis, SUI
Justine Henin, BE
Serena williams, USA
Kim Clijsters, BE
Yanina Wickmayer, BE
student@linux:~/test42$ cat tennis.txt | sed '/BE/d'
Venus Williams, USA
Martina Hingis, SUI
Serena williams, USA
```

11.12. pipe examples

11.12.1. who | wc

How many users are logged on to this system ?

```
[student@linux pipes]$ who
root     tty1         Jul 25 10:50
paul     pts/0        Jul 25 09:29 (laika)
Harry    pts/1        Jul 25 12:26 (barry)
paul     pts/2        Jul 25 12:26 (pasha)
[student@linux pipes]$ who | wc -l
4
```

11.12.2. who | cut | sort

Display a sorted list of logged on users.

```
[student@linux pipes]$ who | cut -d' ' -f1 | sort
Harry
paul
paul
root
```

Display a sorted list of logged on users, but every user only once .

```
[student@linux pipes]$ who | cut -d' ' -f1 | sort | uniq
Harry
paul
root
```

11.12.3. grep | cut

Display a list of all bash `user accounts` on this computer. Users accounts are explained in detail later.

```
student@linux:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
paul:x:1000:1000:paul,,,:/home/paul:/bin/bash
serena:x:1001:1001::/home/serena:/bin/bash
student@linux:~$ grep bash /etc/passwd | cut -d: -f1
root
paul
serena
```

11.13. practice: filters

1. Put a sorted list of all bash users in bashusers.txt.

2. Put a sorted list of all logged on users in onlineusers.txt.

3. Make a list of all filenames in `/etc` that contain the string `conf` in their filename.

4. Make a sorted list of all files in `/etc` that contain the case insensitive string `conf` in their filename.

5. Look at the output of `/sbin/ifconfig`. Write a line that displays only ip address and the subnet mask.

6. Write a line that removes all non-letters from a stream.

7. Write a line that receives a text file, and outputs all words on a separate line.

8. Write a spell checker on the command line. (There may be a dictionary in `/usr/share/dict/` .)

11.14. solution: filters

1. Put a sorted list of all bash users in bashusers.txt.

```
grep bash /etc/passwd | cut -d: -f1 | sort > bashusers.txt
```

2. Put a sorted list of all logged on users in onlineusers.txt.

```
who | cut -d' ' -f1 | sort > onlineusers.txt
```

3. Make a list of all filenames in `/etc` that contain the string `conf` in their filename.

```
ls /etc | grep conf
```

4. Make a sorted list of all files in `/etc` that contain the case insensitive string `conf` in their filename.

```
ls /etc | grep -i conf | sort
```

5. Look at the output of `/sbin/ifconfig`. Write a line that displays only ip address and the subnet mask.

```
/sbin/ifconfig | head -2 | grep 'inet ' | tr -s ' ' | cut -d' ' -f3,5
```

6. Write a line that removes all non-letters from a stream.

```
student@linux:~$ cat text
This is, yes really! , a text with ?&* too many str$ange# characters ;-)
student@linux:~$ cat text | tr -d ',!$?.*&^%#@;()-'
This is yes really  a text with  too many strange characters
```

7. Write a line that receives a text file, and outputs all words on a separate line.

```
student@linux:~$ cat text2
it is very cold today without the sun

student@linux:~$ cat text2 | tr ' ' '\n'
it
is
very
cold
today
without
the
sun
```

8. Write a spell checker on the command line. (There may be a dictionary in `/usr/share/dict/` .)

```
student@linux ~$ echo "The zun is shining today" > text

student@linux ~$ cat > DICT
is
shining
sun
the
today

student@linux ~$ cat text | tr 'A-Z ' 'a-z\n' | sort | uniq | comm -23 - DICT
zun
```

You could also add the solution from question number 6 to remove non-letters, and `tr -s ' '` to remove redundant spaces.

12. regular expressions

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

`Regular expressions` are a very powerful tool in Linux. They can be used with a variety of programs like bash, vi, rename, grep, sed, and more.

This chapter introduces you to the basics of `regular expressions`.

12.1. regex versions

There are three different versions of regular expression syntax:

```
BRE: Basic Regular Expressions
ERE: Extended Regular Expressions
PRCE: Perl Regular Expressions
```

Depending on the tool being used, one or more of these syntaxes can be used.

For example the `grep` tool has the `-E` option to force a string to be read as ERE while `-G` forces BRE and `-P` forces PRCE.

Note that `grep` also has `-F` to force the string to be read literally.

The `sed` tool also has options to choose a regex syntax.

`Read the manual of the tools you use!`

12.2. grep

12.2.1. print lines matching a pattern

`grep` is a popular Linux tool to search for lines that match a certain pattern. Below are some examples of the simplest `regular expressions`.

This is the contents of the test file. This file contains three lines (or three `newline` characters).

```
student@linux:~$ cat names
Tania
Laura
Valentina
```

When `grepping` for a single character, only the lines containing that character are returned.

```
student@linux:~$ grep u names
Laura
student@linux:~$ grep e names
Valentina
student@linux:~$ grep i names
Tania
Valentina
```

The pattern matching in this example should be very straightforward; if the given character occurs on a line, then `grep` will return that line.

12.2.2. concatenating characters

Two concatenated characters will have to be concatenated in the same way to have a match.

This example demonstrates that `ia` will match Tan`ia` but not Valent`ina` and `in` will match Valent`ina` but not Tan`ia`.

```
student@linux:~$ grep a names
Tania
Laura
Valentina
student@linux:~$ grep ia names
Tania
student@linux:~$ grep in names
Valentina
student@linux:~$
```

12.2.3. one or the other

PRCE and ERE both use the pipe symbol to signify OR. In this example we `grep` for lines containing the letter i or the letter a.

```
student@linux:~$ cat list
Tania
Laura
student@linux:~$ grep -E 'i|a' list
Tania
Laura
```

Note that we use the `-E` switch of grep to force interpretion of our string as an ERE.

We need to `escape` the pipe symbol in a BRE to get the same logical OR.

```
student@linux:~$ grep -G 'i|a' list
student@linux:~$ grep -G 'i\|a' list
Tania
Laura
```

12.2.4. one or more

The * signifies zero, one or more occurences of the previous and the + signifies one or more of the previous.

```
student@linux:~$ cat list2
ll
lol
lool
loool
student@linux:~$ grep -E 'o*' list2
ll
lol
lool
loool
student@linux:~$ grep -E 'o+' list2
lol
lool
loool
student@linux:~$
```

12.2.5. match the end of a string

For the following examples, we will use this file.

```
student@linux:~$ cat names
Tania
Laura
Valentina
Fleur
Floor
```

The two examples below show how to use the `dollar character` to match the end of a string.

```
student@linux:~$ grep a$ names
Tania
Laura
Valentina
student@linux:~$ grep r$ names
Fleur
Floor
```

12.2.6. match the start of a string

The `caret character (^)` will match a string at the start (or the beginning) of a line.

Given the same file as above, here are two examples.

```
student@linux:~$ grep ^Val names
Valentina
student@linux:~$ grep ^F names
Fleur
Floor
```

Both the dollar sign and the little hat are called `anchors` in a regex.

12.2.7. separating words

Regular expressions use a `\b` sequence to reference a word separator. Take for example this file:

```
student@linux:~$ cat text
The governer is governing.
The winter is over.
Can you get over there?
```

Simply grepping for `over` will give too many results.

```
student@linux:~$ grep over text
The governer is governing.
The winter is over.
Can you get over there?
```

Surrounding the searched word with spaces is not a good solution (because other characters can be word separators). This screenshot below show how to use `\b` to find only the searched word:

```
student@linux:~$ grep '\bover\b' text
The winter is over.
Can you get over there?
student@linux:~$
```

Note that `grep` also has a `-w` option to grep for words.

```
student@linux:~$ cat text
The governer is governing.
The winter is over.
Can you get over there?
student@linux:~$ grep -w over text
The winter is over.
Can you get over there?
student@linux:~$
```

12.2.8. grep features

Sometimes it is easier to combine a simple regex with `grep` options, than it is to write a more complex regex. These options where discussed before:

```
grep -i
grep -v
grep -w
grep -A5
grep -B5
grep -C5
```

12.2.9. preventing shell expansion of a regex

The dollar sign is a special character, both for the regex and also for the shell (remember variables and embedded shells). Therefore it is advised to always quote the regex, this prevents shell expansion.

```
student@linux:~$ grep 'r$' names
Fleur
Floor
```

12.3. rename

12.3.1. the rename command

On Debian Linux the `/usr/bin/rename` command is a link to `/usr/bin/prename` installed by the `perl` package.

```
student@linux ~ $ dpkg -S $(readlink -f $(which rename))
perl: /usr/bin/prename
```

Red Hat derived systems do not install the same `rename` command, so this section does not describe `rename` on Red Hat (unless you copy the perl script manually).

`There is often confusion on the internet about the rename command because solutions that work fine in Debian (and Ubuntu, xubuntu, Mint, ...) cannot be used in Red Hat (and CentOS, Fedora, ...).`

12.3.2. perl

The `rename` command is actually a perl script that uses `perl regular expressions`. The complete manual for these can be found by typing `perldoc perlrequick` (after installing `perldoc`).

```
root@linux:~# aptitude install perl-doc
The following NEW packages will be installed:
  perl-doc
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 8,170 kB of archives. After unpacking 13.2 MB will be used.
Get: 1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main perl-do ...
Fetched 8,170 kB in 19s (412 kB/s)
Selecting previously unselected package perl-doc.
(Reading database ... 67121 files and directories currently installed.)
Unpacking perl-doc (from .../perl-doc_5.14.2-21+rpi2_all.deb) ...
Adding 'diversion of /usr/bin/perldoc to /usr/bin/perldoc.stub by perl-doc'
Processing triggers for man-db ...
Setting up perl-doc (5.14.2-21+rpi2) ...

root@linux:~# perldoc perlrequick
```

12.3.3. well known syntax

The most common use of the `rename` is to search for filenames matching a certain `string` and replacing this string with an `other string`.

This is often presented as `s/string/other string/` as seen in this example:

```
student@linux ~ $ ls
abc       allfiles.TXT  bllfiles.TXT  Scratch   tennis2.TXT
abc.conf  backup        cllfiles.TXT  temp.TXT  tennis.TXT
student@linux ~ $ rename 's/TXT/text/' *
student@linux ~ $ ls
abc       allfiles.text  bllfiles.text  Scratch    tennis2.text
abc.conf  backup         cllfiles.text  temp.text  tennis.text
```

And here is another example that uses `rename` with the well know syntax to change the extensions of the same files once more:

```
student@linux ~ $ ls
abc       allfiles.text  bllfiles.text  Scratch    tennis2.text
abc.conf  backup         cllfiles.text  temp.text  tennis.text
student@linux ~ $ rename 's/text/txt/' *.text
student@linux ~ $ ls
abc       allfiles.txt  bllfiles.txt  Scratch   tennis2.txt
abc.conf  backup        cllfiles.txt  temp.txt  tennis.txt
student@linux ~ $
```

These two examples appear to work because the strings we used only exist at the end of the filename. Remember that file extensions have no meaning in the bash shell.

The next example shows what can go wrong with this syntax.

```
student@linux ~ $ touch atxt.txt
student@linux ~ $ rename 's/txt/problem/' atxt.txt
student@linux ~ $ ls
abc       allfiles.txt  backup        cllfiles.txt  temp.txt     tennis.txt
abc.conf  aproblem.txt  bllfiles.txt  Scratch       tennis2.txt
student@linux ~ $
```

Only the first occurrence of the searched string is replaced.

12.3.4. a global replace

The syntax used in the previous example can be described as `s/regex/replacement/`. This is simple and straightforward, you enter a `regex` between the first two slashes and a `replacement string` between the last two.

This example expands this syntax only a little, by adding a `modifier`.

```
student@linux ~ $ rename -n 's/TXT/txt/g' aTXT.TXT
aTXT.TXT renamed as atxt.txt
student@linux ~ $
```

The syntax we use now can be described as `s/regex/replacement/g` where s signifies `switch` and g stands for `global`.

Note that this example used the `-n` switch to show what is being done (instead of actually renaming the file).

12.3.5. case insensitive replace

Another `modifier` that can be useful is `i`. this example shows how to replace a case insensitive string with another string.

```
student@linux:~/files$ ls
file1.text  file2.TEXT  file3.txt
student@linux:~/files$ rename 's/.text/.txt/i' *
student@linux:~/files$ ls
file1.txt  file2.txt  file3.txt
student@linux:~/files$
```

12.3.6. renaming extensions

Command line Linux has no knowledge of MS-DOS like extensions, but many end users and graphical application do use them.

Here is an example on how to use `rename` to only rename the file extension. It uses the dollar sign to mark the ending of the filename.

```
student@linux ~ $ ls *.txt
allfiles.txt  bllfiles.txt  cllfiles.txt  really.txt.txt  temp.txt  tennis.txt
student@linux ~ $ rename 's/.txt$/.TXT/' *.txt
student@linux ~ $ ls *.TXT
allfiles.TXT  bllfiles.TXT    cllfiles.TXT    really.txt.TXT
temp.TXT      tennis.TXT
student@linux ~ $
```

Note that the `dollar sign` in the regex means `at the end`. Without the dollar sign this command would fail on the really.txt.txt file.

12.4. sed

12.4.1. stream editor

The `stream editor` or short `sed` uses `regex` for stream editing.

In this example `sed` is used to replace a string.

```
echo Sunday | sed 's/Sun/Mon/'
Monday
```

The slashes can be replaced by a couple of other characters, which can be handy in some cases to improve readability.

```
echo Sunday | sed 's:Sun:Mon:'
Monday
echo Sunday | sed 's_Sun_Mon_'
Monday
echo Sunday | sed 's|Sun|Mon|'
Monday
```

12.4.2. interactive editor

While `sed` is meant to be used in a stream, it can also be used interactively on a file.

```
student@linux:~/files$ echo Sunday > today
student@linux:~/files$ cat today
Sunday
student@linux:~/files$ sed -i 's/Sun/Mon/' today
student@linux:~/files$ cat today
Monday
```

12.4.3. simple back referencing

The `ampersand` character can be used to reference the searched (and found) string.

In this example the `ampersand` is used to double the occurence of the found string.

```
echo Sunday | sed 's/Sun/&&/'
SunSunday
echo Sunday | sed 's/day/&&/'
Sundayday
```

12.4.4. back referencing

Parentheses (often called round brackets) are used to group sections of the regex so they can leter be referenced.

Consider this simple example:

```
student@linux:~$ echo Sunday | sed 's_\(Sun\)_\1ny_'
Sunnyday
student@linux:~$ echo Sunday | sed 's_\(Sun\)_\1ny \1_'
Sunny Sunday
```

12.4.5. a dot for any character

In a `regex` a simple dot can signify any character.

```
student@linux:~$ echo 2014-04-01 | sed 's/....-..-../YYYY-MM-DD/'
YYYY-MM-DD
student@linux:~$ echo abcd-ef-gh | sed 's/....-..-../YYYY-MM-DD/'
YYYY-MM-DD
```

12.4.6. multiple back referencing

When more than one pair of `parentheses` is used, each of them can be referenced separately by consecutive numbers.

```
student@linux:~$ echo 2014-04-01 | sed 's/\(....\)-\(..\)-\(..\)/\1+\2+\3/'
2014+04+01
student@linux:~$ echo 2014-04-01 | sed 's/\(....\)-\(..\)-\(..\)/\3:\2:\1/'
01:04:2014
```

This feature is called `grouping`.

12.4.7. white space

The \s can refer to white space such as a space or a tab.

This example looks for white spaces (\s) globally and replaces them with 1 space.

```
student@linux:~$ echo -e 'today\tis\twarm'
today   is      warm
student@linux:~$ echo -e 'today\tis\twarm' | sed 's_\s_ _g'
today is warm
```

12.4.8. optional occurrence

A question mark signifies that the previous is `optional`.

The example below searches for three consecutive letter o, but the third o is optional.

```
student@linux:~$ cat list2
ll
lol
lool
loool
student@linux:~$ grep -E 'ooo?' list2
lool
loool
student@linux:~$ cat list2 | sed 's/ooo\?/A/'
ll
lol
lAl
lAl
```

12.4.9. exactly n times

You can demand an exact number of times the oprevious has to occur.

This example wants exactly three o's.

```
student@linux:~$ cat list2
ll
lol
lool
loool
student@linux:~$ grep -E 'o{3}' list2
loool
student@linux:~$ cat list2 | sed 's/o\{3\}/A/'
ll
lol
lool
lAl
student@linux:~$
```

12.4.10. between n and m times

And here we demand exactly from minimum 2 to maximum 3 times.

```
student@linux:~$ cat list2
ll
lol
lool
loool
student@linux:~$ grep -E 'o{2,3}' list2
lool
loool
student@linux:~$ grep 'o\{2,3\}' list2
lool
loool
student@linux:~$ cat list2 | sed 's/o\{2,3\}/A/'
ll
lol
lAl
lAl
student@linux:~$
```

12.5. bash history

The `bash shell` can also interprete some regular expressions.

This example shows how to manipulate the exclamation mask history feature of the bash shell.

```
student@linux:~$ mkdir hist
student@linux:~$ cd hist/
student@linux:~/hist$ touch file1 file2 file3
student@linux:~/hist$ ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
student@linux:~/hist$ !l
ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
student@linux:~/hist$ !l:s/1/3
ls -l file3
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file3
student@linux:~/hist$
```

This also works with the history numbers in bash.

```
student@linux:~/hist$ history 6
 2089  mkdir hist
 2090  cd hist/
 2091  touch file1 file2 file3
 2092  ls -l file1
 2093  ls -l file3
 2094  history 6
student@linux:~/hist$ !2092
ls -l file1
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file1
student@linux:~/hist$ !2092:s/1/2
ls -l file2
```

```
-rw-r--r-- 1 paul paul 0 Apr 15 22:07 file2
student@linux:~/hist$
```

13. file globbing

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

Typing `man 7 glob` (on Debian) will tell you that long ago there was a program called `/etc/glob` that would expand wildcard patterns.

Today the shell is responsible for `file globbing` (or dynamic filename generation). This chapter will explain `file globbing`.

13.1. * asterisk

The asterisk `*` is interpreted by the shell as a sign to generate filenames, matching the asterisk to any combination of characters (even none). When no path is given, the shell will use filenames in the current directory. See the man page of `glob(7)` for more information. (This is part of LPI topic 1.103.3.)

```
[student@linux gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[student@linux gen]$ ls File*
File4  File55  FileA  Fileab  FileAB
[student@linux gen]$ ls file*
file1  file2  file3  fileab  fileabc
[student@linux gen]$ ls *ile55
File55
[student@linux gen]$ ls F*ile55
File55
[student@linux gen]$ ls F*55
File55
[student@linux gen]$
```

13.2. ? question mark

Similar to the asterisk, the question mark `?` is interpreted by the shell as a sign to generate filenames, matching the question mark with exactly one character.

```
[student@linux gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[student@linux gen]$ ls File?
File4  FileA
[student@linux gen]$ ls Fil?4
File4
[student@linux gen]$ ls Fil??
File4  FileA
[student@linux gen]$ ls File??
File55  Fileab  FileAB
[student@linux gen]$
```

13.3. [] square brackets

The square bracket [is interpreted by the shell as a sign to generate filenames, matching any of the characters between [and the first subsequent]. The order in this list between the brackets is not important. Each pair of brackets is replaced by exactly one character.

```
[student@linux gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[student@linux gen]$ ls File[5A]
FileA
[student@linux gen]$ ls File[A5]
FileA
[student@linux gen]$ ls File[A5][5b]
File55
[student@linux gen]$ ls File[a5][5b]
File55  Fileab
[student@linux gen]$ ls File[a5][5b][abcdefghijklm]
ls: File[a5][5b][abcdefghijklm]: No such file or directory
[student@linux gen]$ ls file[a5][5b][abcdefghijklm]
fileabc
[student@linux gen]$
```

You can also exclude characters from a list between square brackets with the exclamation mark !. And you are allowed to make combinations of these `wild cards`.

```
[student@linux gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[student@linux gen]$ ls file[a5][!Z]
fileab
[student@linux gen]$ ls file[!5]*
file1  file2  file3  fileab  fileabc
[student@linux gen]$ ls file[!5]?
fileab
[student@linux gen]$
```

13.4. a-z and 0-9 ranges

The bash shell will also understand ranges of characters between brackets.

```
[student@linux gen]$ ls
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[student@linux gen]$ ls file[a-z]*
fileab  fileab2  fileabc
[student@linux gen]$ ls file[0-9]
file1  file2  file3
[student@linux gen]$ ls file[a-z][a-z][0-9]*
fileab2
[student@linux gen]$
```

13.5. $LANG and square brackets

But, don't forget the influence of the `LANG` variable. Some languages include lower case letters in an upper case range (and vice versa).

```
student@linux:~/test$ ls [A-Z]ile?
file1  file2  file3  File4
student@linux:~/test$ ls [a-z]ile?
file1  file2  file3  File4
student@linux:~/test$ echo $LANG
en_US.UTF-8
student@linux:~/test$ LANG=C
student@linux:~/test$ echo $LANG
C
student@linux:~/test$ ls [a-z]ile?
file1  file2  file3
student@linux:~/test$ ls [A-Z]ile?
File4
student@linux:~/test$
```

If `$LC_ALL` is set, then this will also need to be reset to prevent file globbing.

13.6. preventing file globbing

The screenshot below should be no surprise. The `echo *` will echo a * when in an empty directory. And it will echo the names of all files when the directory is not empty.

```
student@linux:~$ mkdir test42
student@linux:~$ cd test42
student@linux:~/test42$ echo *
*
student@linux:~/test42$ touch file42 file33
student@linux:~/test42$ echo *
file33 file42
```

Globbing can be prevented using quotes or by escaping the special characters, as shown in this screenshot.

```
student@linux:~/test42$ echo *
file33 file42
student@linux:~/test42$ echo \*
*
student@linux:~/test42$ echo '*'
*
student@linux:~/test42$ echo "*"
*
```

13.7. practice: shell globbing

1. Create a test directory and enter it.

2. Create the following files :

```
file1
file10
file11
file2
File2
File3
file33
fileAB
filea
fileA
fileAAA
file(
file 2
```

(the last one has 6 characters including a space)

3. List (with ls) all files starting with file

4. List (with ls) all files starting with File

5. List (with ls) all files starting with file and ending in a number.

6. List (with ls) all files starting with file and ending with a letter

7. List (with ls) all files starting with File and having a digit as fifth character.

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

9. List (with ls) all files starting with a letter and ending in a number.

10. List (with ls) all files that have exactly five characters.

11. List (with ls) all files that start with f or F and end with 3 or A.

12. List (with ls) all files that start with f have i or R as second character and end in a number.

13. List all files that do not start with the letter F.

14. Copy the value of $LANG to $MyLANG.

15. Show the influence of $LANG in listing A-Z or a-z ranges.

16. You receive information that one of your servers was cracked, the cracker probably replaced the `ls` command. You know that the `echo` command is safe to use. Can `echo` replace `ls` ? How can you list the files in the current directory with `echo` ?

17. Is there another command besides cd to change directories ?

13.8. solution: shell globbing

1. Create a test directory and enter it.

```
mkdir testdir; cd testdir
```

2. Create the following files :

```
file1
file10
file11
file2
File2
File3
file33
fileAB
filea
fileA
fileAAA
file(
file 2
```

(the last one has 6 characters including a space)

```
touch file1 file10 file11 file2 File2 File3
touch file33 fileAB filea fileA fileAAA
touch "file("
touch "file 2"
```

3. List (with ls) all files starting with file

```
ls file*
```

4. List (with ls) all files starting with File

```
ls File*
```

5. List (with ls) all files starting with file and ending in a number.

```
ls file*[0-9]
```

6. List (with ls) all files starting with file and ending with a letter

```
ls file*[a-z]
```

7. List (with ls) all files starting with File and having a digit as fifth character.

```
ls File[0-9]*
```

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

```
ls File[0-9]
```

9. List (with ls) all files starting with a letter and ending in a number.

```
ls [a-z]*[0-9]
```

10. List (with ls) all files that have exactly five characters.

```
ls ?????
```

13. file globbing

11. List (with ls) all files that start with f or F and end with 3 or A.

```
ls [fF]*[3A]
```

12. List (with ls) all files that start with f have i or R as second character and end in a number.

```
ls f[iR]*[0-9]
```

13. List all files that do not start with the letter F.

```
ls [!F]*
```

14. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

15. Show the influence of $LANG in listing A-Z or a-z ranges.

```
see example in book
```

16. You receive information that one of your servers was cracked, the cracker probably replaced the `ls` command. You know that the `echo` command is safe to use. Can `echo` replace `ls` ? How can you list the files in the current directory with `echo` ?

```
echo *
```

17. Is there another command besides cd to change directories ?

```
pushd popd
```

14. shell variables

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

In this chapter we learn to manage environment `variables` in the shell. These `variables` are often needed by applications.

14.1. $ dollar sign

Another important character interpreted by the shell is the dollar sign `$`. The shell will look for an `environment variable` named like the string following the `dollar sign` and replace it with the value of the variable (or with nothing if the variable does not exist).

These are some examples using $HOSTNAME, $USER, $UID, $SHELL, and $HOME.

```
[student@linux ~]$ echo This is the $SHELL shell
This is the /bin/bash shell
[student@linux ~]$ echo This is $SHELL on computer $HOSTNAME
This is /bin/bash on computer RHELv8u3.localdomain
[student@linux ~]$ echo The userid of $USER is $UID
The userid of paul is 500
[student@linux ~]$ echo My homedir is $HOME
My homedir is /home/paul
```

14.2. case sensitive

This example shows that shell variables are case sensitive!

```
[student@linux ~]$ echo Hello $USER
Hello paul
[student@linux ~]$ echo Hello $user
Hello
```

14.3. creating variables

This example creates the variable `$MyVar` and sets its value. It then uses `echo` to verify the value.

```
[student@linux gen]$ MyVar=555
[student@linux gen]$ echo $MyVar
555
[student@linux gen]$
```

14.4. quotes

Notice that double quotes still allow the parsing of variables, whereas single quotes prevent this.

```
[student@linux ~]$ MyVar=555
[student@linux ~]$ echo $MyVar
555
[student@linux ~]$ echo "$MyVar"
555
[student@linux ~]$ echo '$MyVar'
$MyVar
```

The bash shell will replace variables with their value in double quoted lines, but not in single quoted lines.

```
student@linux:~$ city=Burtonville
student@linux:~$ echo "We are in $city today."
We are in Burtonville today.
student@linux:~$ echo 'We are in $city today.'
We are in $city today.
```

14.5. set

You can use the `set` command to display a list of environment variables. On Ubuntu and Debian systems, the `set` command will also list shell functions after the shell variables. Use `set | more` to see the variables then.

14.6. unset

Use the `unset` command to remove a variable from your shell environment.

```
[student@linux ~]$ MyVar=8472
[student@linux ~]$ echo $MyVar
8472
[student@linux ~]$ unset MyVar
[student@linux ~]$ echo $MyVar

[student@linux ~]$
```

14.7. $PS1

The `$PS1` variable determines your shell prompt. You can use backslash escaped special characters like `\u` for the username or `\w` for the working directory. The `bash` manual has a complete reference.

In this example we change the value of `$PS1` a couple of times.

```
student@linux:~$ PS1=prompt
prompt
promptPS1='prompt '
prompt
prompt PS1='> '
>
> PS1='\u@\h$ '
student@linux$
student@linux$ PS1='\u@\h:\W$'
student@linux:~$
```

To avoid unrecoverable mistakes, you can set normal user prompts to green and the root prompt to red. Add the following to your `.bashrc` for a green user prompt:

```
# color prompt by paul
RED='\[\033[01;31m\]'
WHITE='\[\033[01;00m\]'
GREEN='\[\033[01;32m\]'
BLUE='\[\033[01;34m\]'
export PS1="${debian_chroot:+($debian_chroot)}$GREEN\u$WHITE@$BLUE\h$WHITE\w\$ "
```

14.8. $PATH

The `$PATH` variable is determines where the shell is looking for commands to execute (unless the command is builtin or aliased). This variable contains a list of directories, separated by colons.

```
[[student@linux ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
```

The shell will not look in the current directory for commands to execute! (Looking for executables in the current directory provided an easy way to hack PC-DOS computers). If you want the shell to look in the current directory, then add a . at the end of your $PATH.

```
[student@linux ~]$ PATH=$PATH:.
[student@linux ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
[student@linux ~]$
```

Your path might be different when using su instead of `su -` because the latter will take on the environment of the target user. The root user typically has `/sbin` directories added to the $PATH variable.

```
[student@linux ~]$ su
Password:
[root@linux paul]# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[root@linux paul]# exit
[student@linux ~]$ su -
Password:
[root@linux ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:
[root@linux ~]#
```

14.9. env

The `env` command without options will display a list of `exported variables`. The difference with `set` with options is that `set` lists all variables, including those not exported to child shells.

But `env` can also be used to start a clean shell (a shell without any inherited environment). The `env -i` command clears the environment for the subshell.

Notice in this screenshot that `bash` will set the `$SHELL` variable on startup.

```
[student@linux ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/paul paul
[student@linux ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
/bin/bash
[student@linux ~]$
```

You can use the `env` command to set the `$LANG`, or any other, variable for just one instance of `bash` with one command. The example below uses this to show the influence of the `$LANG` variable on file globbing (see the chapter on file globbing).

```
[student@linux test]$ env LANG=C bash -c 'ls File[a-z]'
Filea  Fileb
[student@linux test]$ env LANG=en_US.UTF-8 bash -c 'ls File[a-z]'
Filea  FileA  Fileb  FileB
[student@linux test]$
```

14.10. export

You can export shell variables to other shells with the `export` command. This will export the variable to child shells.

```
[student@linux ~]$ var3=three
[student@linux ~]$ var4=four
[student@linux ~]$ export var4
[student@linux ~]$ echo $var3 $var4
three four
[student@linux ~]$ bash
[student@linux ~]$ echo $var3 $var4
four
```

But it will not export to the parent shell (previous screenshot continued).

```
[student@linux ~]$ export var5=five
[student@linux ~]$ echo $var3 $var4 $var5
four five
[student@linux ~]$ exit
exit
[student@linux ~]$ echo $var3 $var4 $var5
three four
[student@linux ~]$
```

14.11. delineate variables

Until now, we have seen that bash interprets a variable starting from a dollar sign, continuing until the first occurrence of a non-alphanumeric character that is not an underscore. In some situations, this can be a problem. This issue can be resolved with curly braces like in this example.

```
[student@linux ~]$ prefix=Super
[student@linux ~]$ echo Hello $prefixman and $prefixgirl
Hello  and
[student@linux ~]$ echo Hello ${prefix}man and ${prefix}girl
Hello Superman and Supergirl
[student@linux ~]$
```

14.12. unbound variables

The example below tries to display the value of the `$MyVar` variable, but it fails because the variable does not exist. By default the shell will display nothing when a variable is unbound (does not exist).

```
[student@linux gen]$ echo $MyVar

[student@linux gen]$
```

There is, however, the `nounset` shell option that you can use to generate an error when a variable does not exist.

```
student@linux:~$ set -u
student@linux:~$ echo $Myvar
bash: Myvar: unbound variable
student@linux:~$ set +u
student@linux:~$ echo $Myvar

student@linux:~$
```

In the bash shell `set -u` is identical to `set -o nounset` and likewise `set +u` is identical to `set +o nounset`.

14.13. practice: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)
2. Create a variable `answer` with a value of `42`.
3. Copy the value of $LANG to $MyLANG.
4. List all current shell variables.
5. List all exported shell variables.
6. Do the `env` and `set` commands display your variable ?
6. Destroy your `answer` variable.
7. Create two variables, and `export` one of them.

8. Display the exported variable in an interactive child shell.

9. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use `echo` and the two variables to echo Dumbledore.

10. Find the list of backslash escaped characters in the manual of bash. Add the time to your `PS1` prompt.

14.14. solution: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

```
echo Hello $USER
```

2. Create a variable `answer` with a value of `42`.

```
answer=42
```

3. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

4. List all current shell variables.

```
set
```

```
set|more on Ubuntu/Debian
```

5. List all exported shell variables.

```
env
export
declare -x
```

6. Do the `env` and `set` commands display your variable ?

```
env | more
set | more
```

6. Destroy your `answer` variable.

```
unset answer
```

7. Create two variables, and `export` one of them.

```
var1=1; export var2=2
```

8. Display the exported variable in an interactive child shell.

```
bash
echo $var2
```

9. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use `echo` and the two variables to echo Dumbledore.

```
varx=Dumb; vary=do

echo ${varx}le${vary}re
solution by Yves from Dexia : echo $varx'le'$vary're'
solution by Erwin from Telenet : echo "$varx"le"$vary"re
```

10. Find the list of backslash escaped characters in the manual of bash. Add the time to your PS1 prompt.

```
PS1='\t \u@\h \W$ '
```

15. introduction to scripting

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/, Bert Van Vreckem https://github.com/bertvv/)

The goal of this chapter is to give you all the information in order to read, write and understand small, long and complex shell scripts.

You should have read and understood `part III shell expansion` and `part IV pipes and commands` before starting this chapter.

15.1. introduction

When you open a terminal and type a command, you are using a *shell*, an interactive environment that interprets your commands, executes them, and shows you the output the command generates. Most Linux distributions have Bash (the "Bourne Again Shell") as the default, but there are others as well: the original "Bourne shell" (`sh`), the "Debian Amquist Shell" (`dash`, a modern implementation of `sh`), the "Korn shell" (`ksh`), the "C shell" (`csh`), and the "Z shell" (`zsh`), to name a few.

A sequence of commands can be saved in a file and executed as a single command. This is called a *script*. Shell scripts are used to automate tasks, and are an essential tool for system administrators and developers. Subsequently, this means that system administrators or SysOps also need solid knowledge of *scripting* to understand how their servers and their applications are started, updated, upgraded, patched, maintained, configured and removed, and also to understand how a user environment is built.

Shells have also support for programming constructs (like loops, functions, variables, etc.) so that you can write more complex scripts. This makes a scripting language basically as powerful as a programming language. Scripting languages are often interpreted, rather than compiled.

If you copy a script to one of the `bin` directories (e.g. `/usr/local/bin`), you can execute it from the command line just like any other command. In fact, many UNIX/Linux commands are essentially `scripts`. You can check this for yourself by executing the `file` command on the executables in the `/bin` directory. For example:

```
student@linux:~$ file /usr/bin/* | awk '{ print($2, $3, $4) }' \
  | sort | uniq -c | sort -nr
    466 ELF 64-bit LSB
    168 symbolic link to
     74 POSIX shell script,
     71 Perl script text
     14 Python script, ASCII
     10 setuid ELF 64-bit
      7 setgid ELF 64-bit
      6 Bourne-Again shell script,
      2 Python script, Unicode
      1 Python script, ISO-8859
```

We find POSIX (Bourne), Bash, Perl and Python scripts, as well as ELF binaries (compiled programs). This shows that a significant portion of the commands in a typical Linux system are actually scripts.

Bash scripting is a valuable skill for any Linux user, but these days, its applications are no longer limited to Linux. Bash is also present on macOS (albeit an older version), and with the advent of Windows Subsystem for Linux (WSL), Bash is now available for Windows users as well. Moreover, Git Bash, a Bash shell for Windows, is also available.

15.2. hello world

Just like in every programming course, we start with a simple `hello_world` script. The following script will output `Hello World`.

```
1 echo Hello World
```

After creating this simple script in `nano`, `vi`, or with `echo`, you'll have to `chmod +x hello_world` to make it executable. And unless you add the scripts directory to your path, you'll have to type the path to the script for the shell to be able to find it.

```
student@linux:~$ echo echo Hello World > hello_world
student@linux:~$ chmod +x hello_world
student@linux:~$ ./hello_world
Hello World
student@linux:~$
```

15.3. she-bang

Let's expand our example a little further by putting `#!/bin/bash` on the first line of the script. The `#!` is called a `she-bang` (sometimes called `sha-bang`), where the `she-bang` is the first two characters of the script.

Open the file with `nano hello_world` or `vi hello_world` and add the following line at the top of the file.

```
1 #!/bin/bash
2 echo Hello World
```

You can never be sure which (interactive) shell a user is running. A script that works flawlessly in `bash` might not work in `ksh`, `csh`, or `dash`. To instruct a shell to run your script with a specific interpreter, you should start your script with a `she-bang` followed by the absolute path to the executable of the interpreter.

This script will run in a bash shell.

```
1 #!/bin/bash
2 echo -n hello
3 echo A bash subshell $(echo -n hello)
```

This script will be interpreted by Python:

```
1 #!/usr/bin/env python3
2 print("Hello World!")
```

The following script will run in a Korn shell (unless `/bin/ksh` is a hard link to `/bin/bash`). The `/etc/shells` file contains a list of shells available on your system. Check it to see which ones are available to you

```
#!/bin/ksh
echo -n hello
echo a Korn subshell $(echo -n hello)
```

If you're not sure in which `bin` directory the shell executable is located,you can use `env`. The command `env` is normally used to print environment variables, but in the context of a script, it is used to launch the correct interpreter.

```
#!/usr/bin/env bash
echo -n hello
echo A bash subshell $(echo -n hello)
```

This is particularly useful for macOS users: out-of-the-box, a macOS system has a very old version of `bash` in `/bin/bash`. If you want to use a more recent version, you can install it with Homebrew, that will put it in `/usr/local/bin/bash`. If you use `#!/usr/bin/env bash` in your scripts, the newer version will be used.

15.4. comments

When writing Bash scripts, it is always a good practice to make your code clean and easily understandable. Organizing your code in blocks, indenting, giving variables and functions descriptive names are several ways to do this. Another way to improve the readability of your code is by using comments. A comment is a human-readable explanation or annotation that is written in the shell script.

Let's expand our example a little further by adding comment lines.

```
#!/usr/bin/env bash
#
# hello_world.sh -- My first script
#
echo Hello World

# this is old way of calling for subshell with backtick ``
echo A bash subshell `echo -n hello`

# this is more modern way of calling for subshell with dollar and brackets
 ↪  $()
echo A bash subshell $(echo -n hello)

#NOTICE: backtick might not work in future versions of bash shell
```

15.5. extension

A general convention is to give files an extension that indicates the file type. On a Linux system, this is not strictly necessary. Remember that you can always use the `file` command to determine the type of a file by scanning its contents. The system will not care if you call your script `hello_world.sh` or `hello_world`. However, it is a good practice to use an extension, as it makes it easier to identify the type of file.

We recommend to always give your scripts the `.sh` extension, but to remove the extension when you install it in a `bin` directory as a command.

15.6. shell variables

Here is a simple example of a shell variable used inside a script.

```
#!/bin/bash
# hello-user.sh -- example of a shell variable in a script
echo "Hello ${USER}"
```

In Bash, you can access the value of a variable by prefixing the variable name with the $ sign. The braces are not mandatory in this case, but they are a good practice to avoid ambiguity. In some cases they are required, so it's best to be consistent in your coding style.

The variable `${USER}` is a shell variable that is defined by the system when you log in.

```
student@linux:~$ chmod +x hello-user.sh
student@linux:~$ ./hello-user.sh
Hello student
```

15.7. variable assignment

Assigning a variable is done by using the = operator. The variable name must start with a letter or an underscore, and can contain only letters, digits, or underscores. Remark that spaces are not allowed around the = sign!

```
#!/bin/bash
# hello-var.sh -- example of variable assignment
user="Tux"

echo "Hello ${user}"
```

Because variable names are case-sensitive, this variable `${user}` is different from `${USER}` in the previous example!

> **Tip: naming convention.** You can use any name for a variable, but it is a good practice to use all uppercase letters for environment variables (e.g. `${USER}`) and constants and all lowercase letters for local variables (e.g. `${user}`). This is also recommended by the Google Shell Style Guide. If a variable consists of multiple words, use underscores to separate them (e.g. `${current_user}`).

Running the script:

```
student@linux:~$ chmod +x hello-var.sh
student@linux:~$ ./hello-var.sh
Hello Tux
```

Scripts can contain variables, but since scripts are run in their own subshell, the variables do not survive the end of the script.

```
student@linux:~$ echo $user

student@linux:~$ ./hello-var.sh
Hello Tux
student@linux:~$ echo $user

student@linux:~$
```

15.8. unbound variables

Remove the line `user="Tux"` from the script, or comment out the line and run it again. What do you expect to happen if the variable user is not assigned, but we try to use it in the script?

```
student@linux:~$ ./hello-var.sh
Hello
```

Bash will not complain if you use a variable that is not assigned, but it will simply replace the variable with an empty string. This can lead to unexpected results and is a common cause of bugs that can be hard to find. However, you can change the behavior of the shell by starting your scripts with the command `set -o nounset` (or shorter: `set -u`). This will cause the script to exit with an error if you try to use an unassigned variable.

Add the line to the script, right below the comment lines and try again!

```
1 #!/bin/bash
2 # hello-var.sh -- example of variable assignment
3
4 set -o nounset
5
6 echo "Hello ${user}"
```

Running the script:

```
student@linux:~$ ./hello-var.sh
./hello-var.sh: line 6: user: unbound variable
```

This is what you want to see. The script exits with an error, and you can see the line number where the error occurred and which variable is unbound. Start all your scripts with `set -o nounset` to prevent this kind of error!

15.9. sourcing a script

Luckily, you can force a script to run in the same shell; this is called `sourcing` a script.

```
student@linux:~$ source hello-var.sh
Hello Tux
student@linux:~$ echo $name
Tux
```

Instead of `source`, you can use the `.` (dot) command.

```
student@linux:~$ . hello-var.sh
Hello Tux
student@linux:~$ echo $name
Tux
```

15.10. quoting

Go back to `hello-user.sh` and replace the double quotes with single quotes:

```
#!/bin/bash
# hello-user.sh -- example of a shell variable in a script
echo 'Hello ${USER}'
```

Run the script again:

```
student@linux:~$ ./hello-user.sh
Hello ${USER}
```

What happened? By using single quotes, we turned off the shell's variable expansion. The shell will not replace `${USER}` with the value of the `USER` variable. This is why you should use double quotes when you want to use a variable.

Using quotes is important. Most of the times, when you reference the value of a variable, you should enclose it in double quotes. To illustrate this, write the following script:

```
#!/bin/bash
# create-file.sh -- example of using quotes
file="my file.txt"
touch $file
```

What we expect is that the script will create a file called `my file.txt`. However, when we run the script:

```
student@linux:~$ ./create-file.sh
student@linux:~$ ls -l
total 4
-rwxr-xr-x 1 student student     88 Mar  6 16:20 create-file.sh
-rw-r--r-- 1 student student      0 Mar  6 16:20 file.txt
-rw-r--r-- 1 student student      0 Mar  6 16:20 my
```

So actually two files were created, one named `my` and the other `file.txt`. The reason has to do with the way Bash interprets a command and how it substitutes variables. The line

```
touch $file
```

is expanded to

```
touch my file.txt
```

without the quotes. The `touch` command now sees two arguments, `my` and `file.txt`, and creates two files. To fix this, you should always use double quotes:

```
#!/bin/bash
# create-file.sh -- example of using quotes
file="my file.txt"
touch "${file}"
```

Now the expansion of the variable is done within the quotes, and the `touch` command sees only one argument.

```
student@linux:~$ ./create-file.sh
student@linux:~$ ls -l
total 4
-rwxr-xr-x 1 student student     92 Mar  6 16:20 create-file.sh
-rw-r--r-- 1 student student      0 Mar  6 16:20 'my file.txt'
```

15.11. troubleshooting a script

Another way to run a script in a separate shell is by typing `bash` with the name of the script as a parameter. Expanding this to `bash -x` allows you to see the commands that the shell is executing (after shell expansion).

Try this with the `create-file.sh` script! The incorrect version without the quotes:

```
$ bash -x create-file.sh
+ file='my file.txt'
+ touch my file.txt
```

Notice the absence of the commented (#) line, and the replacement of the variable in the argument `touch`.

After the fix, you get:

```
$ bash -x create-file.sh
+ file='my file.txt'
+ touch 'my file.txt'
```

Do you notice the difference?

In longer scripts, this setting produces a lot of output, which may be hard to read. You can limit the output to a specific problematic part of your script by using `set -x` and `set +x` to turn the debugging on and off.

```
#!/bin/bash
# create-file.sh -- example of using quotes
file="my file.txt"

set -x
touch "${file}"
set +x
```

15.12. Bash's "strict mode"

Apart from the `nounset` shell option, there are two other options that are very useful for debugging scripts: `set -o errexit` (or `set -e`) and `set -o pipefail`. The first option causes the script to exit with an error if any command fails. The second option gives better error messages when a command in a pipeline fails.

Start all your scripts with the following lines to prevent errors and to make debugging easier:

```
#!/bin/bash --
set -o nounset
set -o errexit
set -o pipefail
```

This is called "strict mode" by some. You can write this shorter in one line as `set -euo pipefail`, but this is less readable.

15.13. prevent setuid root spoofing

Some user may try to perform `setuid` based script `root spoofing`. This is a rare but possible attack. To improve script security and to avoid interpreter spoofing, you need to add `--` after the `#!/bin/bash`, which disables further option processing so the shell will not accept any options.

```
#!/usr/bin/env bash -
or
#!/usr/bin/env bash --
```

Any arguments after the `--` are treated as filenames and arguments. An argument of `-` is equivalent to `--`.

15.14. practice: introduction to scripting

1. Write a Python "Hello World" script, give it a shebang and make it executable. Execute it like you would a shell script and verify that this works.
2. What would happen if you remove the shebang and try to execute the script again?
3. Create a Bash script `greeting.sh` that says hello to the user (make use of the shell variable with the current user's login name), prints the current date and time, and prints a quote, e.g.:

   ```
   student@linux:~$ ./greeting.sh
   Hello student, today is:
   Wed Mar  6 09:04:19 PM UTC 2024
   Quote of the day:

    ______________________________________
   / Having nothing, nothing can he lose. \
   |                                      |
   \ -- William Shakespeare, "Henry VI"   /
    --------------------------------------
           \   ^__^
            \  (oo)\_______
               (__)\       )\/\
                   ||----w |
                   ||     ||
   ```

   Ensure that you apply the shell settings to make your script easier to debug.
4. Copy the script to `/usr/local/bin` without the extension and verify that you can run it from any directory as a command.
5. Take another look at the script `hello-var.sh` where we printed a variable that was not assigned:

   ```
   #!/bin/bash
   # hello-var.sh -- example of variable assignment
   # user="Tux" # Remark: this line is commented out

   echo "Hello ${user}"
   ```

   What happens if you assign the value `Tux` to the variable `user` on the interactive shell and then run the script? What do we have to do to make sure the variable is available in the script?
6. What if we change the value of the variable `user` in the script? Will this change affect the value of the variable in the interactive shell after the script is finished?

15.15. solution: introduction to scripting

1. Write a Python Hello World script, give it a shebang and make it executable.

```
#!/usr/bin/python3
print("Hello, World!")
```

```
$ chmod +x hello.py
$ ./hello.py
Hello, World!
```

2. What would happen if you remove the shebang and try to execute the script again?

   The script will be executed by the default interpreter, in this case, the Bash shell, which will not understand the Python syntax.

```
$ ./hello.py
./hello.py: line 1: syntax error near unexpected token `"Hello world!"'
./hello.py: line 1: `print("Hello world!")'
```

3. Create a Bash script `greeting.sh` that says hello to the user (make use of the shell variable with the current user's login name), prints the current date and time, and prints a quote. Ensure that you apply the shell settings to make your script easier to debug.

```
#! /bin/bash --

set -o nounset
set -o errexit
set -o pipefail

echo "Hello ${USER}, today is:"
date
echo "Quote of the day:"
fortune | cowsay
```

4. Copy the script to `/usr/local/bin` without the extension and verify that you can run it from any directory as a command.

```
student@linux:~$ sudo cp greeting.sh /usr/local/bin/greeting
student@linux:~$ greeting
Hello student, today is:
Wed Mar  6 09:17:00 PM UTC 2024
Quote of the day:
 ______________________________________
/ You plan things that you do not even \
| attempt because of your extreme      |
\ caution.                             /
 --------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
student@linux:~$ cd /tmp
student@linux:/tmp$ greeting
Hello student, today is:
Wed Mar  6 09:17:08 PM UTC 2024
Quote of the day:
 _________________________________
< You will be successful in love. >
 ---------------------------------
```

```
 \   ^__^
  \  (oo)\_______
     (__)\       )\/\
         ||----w |
         ||     ||
```

5. Take another look at the script `hello-var.sh` where we printed a variable that was not assigned. What happens if you assign the value `Tux` to the variable `user` on the interactive shell and then run the script? What do we have to do to make sure the variable is available in the script?

   ```
   student@linux:~$ ./hello-var.sh
   Hello
   student@linux:~$ user=Tux
   student@linux:~$ ./hello-var.sh
   Hello
   student@linux:~$ export user
   student@linux:~$ ./hello-var.sh
   Hello Tux
   ```

6. What if we change the value of the variable `user` in the script? Will this change affect the value of the variable in the interactive shell after the script is finished?

   We change the script to:

   ```
   1 #!/bin/bash
   2 # hello-var.sh -- example of variable assignment
   3 user="Linus"
   4
   5 echo "Hello ${user}"
   ```

   And execute it:

   ```
   student@linux:~$ export user=Tux
   student@linux:~$ echo $user
   Tux
   student@linux:~$ ./hello-var.sh
   Hello Linus
   student@linux:~$ echo $user
   Tux
   ```

   The change in the script does not affect the value of the variable in the interactive shell after the script is finished!

Part IV.

Software management; DHCP

16. package management

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/, Bert Van Vreckem https://github.com/bertvv/)

Most Linux distributions have a **package management** system with online **repositories** containing thousands of packages. This makes it very easy to install, update and remove applications, operating system components, documentation and much more.

We first discuss the Debian package format `.deb` and its tools `dpkg`, `apt-get` and `apt`. This should be similar on Debian, Ubuntu, Mint and all derived distributions.

Then we take a look at the Red Hat package format `.rpm` and its tools `rpm` and `dnf`. This should be similar on Red Hat, Fedora, AlmaLinux and all derived distributions.

16.1. package terminology

16.1.1. repository

A lot of software and documentation for your Linux distribution is available as **packages** in one or more centrally distributed **repositories**. The packages in such a repository are tested and very easy to install (or remove) with a graphical or command line installer.

16.1.2. .deb packages

Debian, Ubuntu, Mint and all derivatives of Debian and Ubuntu use `.deb` packages. To manage software on these systems, you can use `apt` or `apt-get`, both these tools are a front end for `dpkg`.

16.1.3. .rpm packages

Red Hat, Fedora, CentOS, OpenSUSE, Mandriva, Red Flag and others use `.rpm` packages. The tools to manage software packages on these systems are `dnf` and `rpm`.

16.1.4. dependency

Some packages need other packages to function. Tools like `apt-get`, `apt` and `dnf` will install all **dependencies** you need. When using `dpkg` or `rpm`, or when building from **source**, you will need to install dependencies yourself.

16.1.5. open source

These repositories contain a lot of independent **open source software**. Often the source code is customized to integrate better with your distribution. Most distributions also offer this modified source code as a **package** in one or more **source repositories**.

You are free to go to the project website itself (samba.org, apache.org, github.com ...) and download the *vanilla* (= without the custom distribution changes) source code.

16.1.6. GUI software management

End users have several graphical applications available via the desktop (look for *add/remove software* or something similar).

Below a screenshot of Ubuntu Software Center running on Ubuntu 12.04. Graphical tools are not discussed in this book.

16.2. deb package management

16.2.1. about deb

Most people use `apt` or `apt-get` (APT = Advanced Package Tool) to manage their Debian/Ubuntu family of Linux distributions. Both are a front end for `dpkg` and are themselves a back end for *synaptic* and other graphical tools.

16.2.2. dpkg -l

The low level tool to work with `.deb` packages is `dpkg`. Among other things, you can use `dpkg` to list all installed packages on a Debian server.

```
student@debian:~$ dpkg -l | wc -l
365
```

Compare this to the same list on a Linux Mint system with a graphical desktop installed.

```
student@mint:~$ dpkg -l | wc -l
2118
```

16.2.3. dpkg -l $package

Here is an example on how to get information on an individual package. The ii at the beginning means the package is installed.

```
root@debian:~# dpkg -l rsync | tail -1 | tr -s  ' '
ii rsync 3.2.7-1 amd64 fast, versatile, remote (and local) file-copying tool
```

16.2.4. dpkg -S

You can find the package responsible for installing a certain file on your computer using `dpkg -S`. This example shows how to find the package for three files on a typical Debian server.

```
student@debian:~$ dpkg -S /usr/share/doc/tmux/ /etc/ssh/ssh_config /sbin/ifconfig
dpkg-query: no path found matching pattern /usr/share/doc/tmux/
openssh-client: /etc/ssh/ssh_config
net-tools: /sbin/ifconfig
```

16.2.5. dpkg -L

In reverse, you can also get a list of all files that have been installed by a certain program. Below is the list for the `curl` package.

```
student@debian:~$ dpkg -L curl
/.
/usr
/usr/bin
/usr/bin/curl
/usr/share
/usr/share/doc
/usr/share/doc/curl
/usr/share/doc/curl/changelog.Debian.gz
/usr/share/doc/curl/changelog.gz
/usr/share/doc/curl/copyright
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/curl.1.gz
/usr/share/zsh
/usr/share/zsh/vendor-completions
/usr/share/zsh/vendor-completions/_curl
```

16.2.6. dpkg

You could use `dpkg -i` to install a package and `dpkg -r` to remove a package, but you'd have to manually download the packge and keep track of dependencies. Using `apt-get` or `apt` is much easier.

16.2.7. apt-get

`Debian` has been using `apt-get` to manage packages since 1998. Today Debian and many Debian-based distributions still actively support `apt-get`, though some experts claim `apt`, released in 2014, is better at handling dependencies than `apt-get`.

Both commands use the same configuration files and can be used alternately; whenever you see `apt-get` in documentation, feel free to type `apt`.

We will start with `apt-get` and discuss `apt` in the next section.

16.2.8. apt-get update

When typing `apt-get update` you are downloading the names, versions and short description of all packages available on all configured repositories for your system. Remark that you need to be root to run this command.

```
student@debian:~$ apt-get update
Reading package lists ... Done
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
student@debian:~$ sudo apt-get update
Hit:1 http://security.debian.org/debian-security bookworm-security InRelease
Hit:2 http://httpredir.debian.org/debian bookworm InRelease
Hit:3 http://httpredir.debian.org/debian bookworm-updates InRelease
Reading package lists ... Done
```

In the example below you can see an interaction with an Ubuntu system. Some repositories are at the url `be.archive.ubuntu.com` because this computer was installed in Belgium. This mirror URL can be different for you.

```
student@ubuntu:~$ sudo apt-get update
Ign http://be.archive.ubuntu.com precise InRelease
Ign http://extras.ubuntu.com precise InRelease
Ign http://security.ubuntu.com precise-security InRelease
Ign http://archive.canonical.com precise InRelease
Ign http://be.archive.ubuntu.com precise-updates InRelease
...
Hit http://be.archive.ubuntu.com precise-backports/main Translation-en
Hit http://be.archive.ubuntu.com precise-backports/multiverse Translation-en
Hit http://be.archive.ubuntu.com precise-backports/restricted Translation-en
Hit http://be.archive.ubuntu.com precise-backports/universe Translation-en
Fetched 13.7 MB in 8s (1682 kB/s)
Reading package lists ... Done
student@ubuntu:~$
```

Tips:

- Run `apt-get update` every time before performing other package operations to ensure your metadata is up-to-date.
- Since the package repositories are hosted on web servers, you can open any repository URL in your browser to see how the repository is structured.

16.2.9. apt-get upgrade

One of the nicest features of `apt-get` is that it allows for a secure update of *all software currently installed* on your computer with just *one* command.

```
student@debian:~$ sudo apt-get upgrade
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

The above transcript shows that all software is updated to the latest version available for my distribution. Below is an example of a system with software that can be updated. Some lines were ommitted for brevity.

```
student@debian:~$ sudo apt-get upgrade
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Calculating upgrade ... Done
The following packages have been kept back:
  linux-image-amd64
The following packages will be upgraded:
   base-files bind9-dnsutils bind9-host bind9-libs cryptsetup cryptsetup-
bin libcryptsetup12 libgnutls30 libnss-systemd libpam-systemd libsystemd-
shared libsystemd0 libudev1 systemd systemd-sysv
  systemd-timesyncd tar tzdata udev usr-is-merged
20 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Need to get 13.0 MB of archives.
After this operation, 75.8 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.debian.org/debian-security bookworm-security/main amd64 bind9-
host amd64 1:9.18.24-1 [305 kB]
[ ... ]
Get:20 http://httpredir.debian.org/debian bookworm/main amd64 cryptsetup amd64 2:2.6.1-
4~deb12u2 [213 kB]
Fetched 13.0 MB in 1s (20.3 MB/s)
Reading changelogs ... Done
Preconfiguring packages ...
(Reading database ... 29205 files and directories currently installed.)
Preparing to unpack .../base-files_12.4+deb12u5_amd64.deb ...
Unpacking base-files (12.4+deb12u5) over (12.4+deb12u4) ...
Setting up base-files (12.4+deb12u5) ...
Installing new version of config file /etc/debian_version ...
[ ... ]
Preparing to unpack .../5-cryptsetup_2%3a2.6.1-4~deb12u2_amd64.deb ...
Unpacking cryptsetup (2:2.6.1-4~deb12u2) over (2:2.6.1-4~deb12u1) ...
Setting up systemd-sysv (252.22-1~deb12u1) ...
[ ... ]
Setting up bind9-dnsutils (1:9.18.24-1) ...
Processing triggers for initramfs-tools (0.142) ...
update-initramfs: Generating /boot/initrd.img-6.1.0-17-amd64
[ ... ]
Processing triggers for mailcap (3.70+nmu1) ...
```

Tip: Have you noticed that almost every time that you update software on Windows, you are asked to reboot your computer? This is **not** the case with Linux! The only time you need to reboot is when you update the kernel.

16.2.10. apt-get clean

`apt-get` keeps a copy of downloaded packages in `/var/cache/apt/archives`, as can be seen in this screenshot.

```
student@debian:~$ ls /var/cache/apt/archives/ | head
base-files_12.4+deb12u5_amd64.deb
bind9-dnsutils_1%3a9.18.24-1_amd64.deb
bind9-host_1%3a9.18.24-1_amd64.deb
bind9-libs_1%3a9.18.24-1_amd64.deb
cryptsetup_2%3a2.6.1-4~deb12u2_amd64.deb
cryptsetup-bin_2%3a2.6.1-4~deb12u2_amd64.deb
libcryptsetup12_2%3a2.6.1-4~deb12u2_amd64.deb
libgnutls30_3.7.9-2+deb12u2_amd64.deb
libnss-systemd_252.22-1~deb12u1_amd64.deb
libpam-systemd_252.22-1~deb12u1_amd64.deb
```

Running `apt-get clean` removes all .deb files from that directory.

```
student@debian:~$ sudo apt-get clean
student@debian:~$ ls /var/cache/apt/archives/*.deb
ls: cannot access /var/cache/apt/archives/*.deb: No such file or directory
```

16.2.11. apt-cache search

Use `apt-cache search` to search for availability of a package. Here we look for `rsync`.

```
student@debian:~$ apt-cache search rsync | grep '^rsync'
rsync - fast, versatile, remote (and local) file-copying tool
rsyncrypto - rsync friendly encryption
```

16.2.12. apt-get install

You can install one or more applications by appending their name behind `apt-get install`. The following example shows how to install the `tftp-hpa` package (a TFTP server).

```
student@debian:~$ sudo apt-get install tftpd-hpa
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Suggested packages:
  pxelinux
The following NEW packages will be installed:
  tftpd-hpa
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 41.9 kB of archives.
After this operation, 117 kB of additional disk space will be used.
Get:1 http://httpredir.debian.org/debian bookworm/main amd64 tftpd-hpa amd64 5.2+20150808-
1.4 [41.9 kB]
Fetched 41.9 kB in 0s (241 kB/s)
Preconfiguring packages ...
Selecting previously unselected package tftpd-hpa.
(Reading database ... 29179 files and directories currently installed.)
Preparing to unpack .../tftpd-hpa_5.2+20150808-1.4_amd64.deb ...
```

```
Unpacking tftpd-hpa (5.2+20150808-1.4) ...
Setting up tftpd-hpa (5.2+20150808-1.4) ...
Processing triggers for man-db (2.11.2-2) ...
```

The `apt-get` command will ask the user to confirm the installation of the package by pressing "y" and ENTER. You can use the `-y` option to automatically answer yes to all questions.

The following example installs the `vim` package (VI iMproved, a powerful text editor for the terminal). **Remark** that some additional packages are installed as dependencies!

```
student@debian:~$ sudo apt-get install -y vim
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libgpm2 libsodium23 vim-runtime
Suggested packages:
  gpm ctags vim-doc vim-scripts
The following NEW packages will be installed:
  libgpm2 libsodium23 vim vim-runtime
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 8,768 kB of archives.
After this operation, 41.5 MB of additional disk space will be used.
[ ... ]
Setting up libsodium23:amd64 (1.0.18-1) ...
Setting up libgpm2:amd64 (1.20.7-10+b1) ...
Setting up vim-runtime (2:9.0.1378-2) ...
Setting up vim (2:9.0.1378-2) ...
[ ... ]
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+deb12u4) ...
```

16.2.13. apt-get remove

You can remove one or more applications by appending their name behind `apt-get remove`.

```
student@debian:~$ sudo apt-get remove tftpd-hpa
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages will be REMOVED:
  tftpd-hpa
0 upgraded, 0 newly installed, 1 to remove and 1 not upgraded.
After this operation, 117 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 29194 files and directories currently installed.)
Removing tftpd-hpa (5.2+20150808-1.4) ...
Processing triggers for man-db (2.11.2-2) ...
```

If we use `dpkg -l` to check the status of the `tftpd-hpa` package, we see that it is removed, some configuration (rc) files are left on the system. Indeed, the configuration file `/etc/init/tftpd-hpa.conf` is not removed! We'll solve this in the next section.

```
student@debian:~$ dpkg -l tftpd-hpa | tail -1
rc  tftpd-hpa      5.2+20150808-1.4 amd64        HPA's tftp server
student@debian:~$ ls -l /etc/init/tftpd-hpa.conf
-rw-r--r-- 1 root root 980 Oct 25  2022 /etc/init/tftpd-hpa.conf
```

The example below shows how to remove the `vim` package. Note that dependencies are **not** removed! You can execute `sudo apt autoremove` afterwards (as is suggested by the output of the command!) to remove those as well.

```
student@debian:~$ sudo apt-get remove vim
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  libsodium23 vim-runtime
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  vim
0 upgraded, 0 newly installed, 1 to remove and 1 not upgraded.
After this operation, 3,738 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 31257 files and directories currently installed.)
Removing vim (2:9.0.1378-2) ...
[ ... ]
student@debian:~$ sudo apt-get autoremove
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages will be REMOVED:
  libsodium23 vim-runtime
0 upgraded, 0 newly installed, 2 to remove and 1 not upgraded.
After this operation, 37.7 MB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 31247 files and directories currently installed.)
Removing libsodium23:amd64 (1.0.18-1) ...
Removing vim-runtime (2:9.0.1378-2) ...
Removing 'diversion of /usr/share/vim/vim90/doc/help.txt to /usr/share/vim/vim90/doc/help.
tiny by vim-runtime'
Removing 'diversion of /usr/share/vim/vim90/doc/tags to /usr/share/vim/vim90/doc/tags.vim-
tiny by vim-runtime'
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+deb12u4) ...
```

16.2.14. apt-get purge

You can purge one or more applications by appending their name behind `apt-get purge`. Purging will also remove all existing configuration files related to that application. The screenshot shows how to purge the `tftpd-hpa` package.

```
student@debian:~$ ls -l /etc/init/tftpd-hpa.conf
-rw-r--r-- 1 root root 980 Oct 25  2022 /etc/init/tftpd-hpa.conf
student@debian:~$ sudo apt-get purge tftpd-hpa
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages will be REMOVED:
```

```
  tftpd-hpa*
0 upgraded, 0 newly installed, 1 to remove and 1 not upgraded.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
(Reading database ... 29182 files and directories currently installed.)
Purging configuration files for tftpd-hpa (5.2+20150808-1.4) ...
student@debian:~$ ls -l /etc/init/tftpd-hpa.conf
ls: cannot access '/etc/init/tftpd-hpa.conf': No such file or directory
```

Note that `dpkg` has no information about a purged package!

```
student@debian:~$ dpkg -l tftpd-hpa | tail -1 | tr -s ' '
dpkg-query: no packages found matching tftpd-hpa
```

16.2.15. apt

Nowadays, most people use `apt` for package management on Debian, Mint and Ubuntu systems. That does not mean that `apt-get` is no longer useful. In scripts, it is actually recommended to use `apt-get` because its options and behaviour are more stable and predictable than `apt`. For interactive use, `apt` is more user-friendly.

To synchronize with the repositories.

```
sudo apt update
```

To patch and upgrade all software to the latest version on Debian.

```
sudo apt upgrade
```

To patch and upgrade all software to the latest version on Ubuntu and Mint.

```
sudo apt safe-upgrade
```

To install an application with all dependencies.

```
sudo apt install $package
```

To search the repositories for applications that contain a certain string in their name or description.

```
apt search $string
```

To remove an application.

```
sudo apt remove $package
```

To remove an application and all configuration files.

```
sudo apt purge $package
```

16.2.16. /etc/apt/sources.list

Both `apt-get` and `apt` use the same configuration information in `/etc/apt/`. The main configuration file is `/etc/apt/sources.list` and the directory `/etc/apt/sources.list.d/` contains additional files. These contain a list of http or ftp sources where packages for the distribution can be downloaded. Third party software vendors may provide their own package repositories for Debian or Ubuntu. These repositories are typically added through a new file in `/etc/apt/sources.list.d/`.

This is what that list looks like on a Debian server system shortly after installation.

```
student@debian:~$ cat /etc/apt/sources.list
deb http://httpredir.debian.org/debian/ bookworm main non-free-firmware
deb-src http://httpredir.debian.org/debian/ bookworm main non-free-firmware

deb http://security.debian.org/debian-security bookworm-security main non-
free-firmware
deb-src http://security.debian.org/debian-security bookworm-security main non-
free-firmware

# bookworm-updates, to get updates before a point release is made;
deb http://httpredir.debian.org/debian/ bookworm-updates main non-free-
firmware
deb-src http://httpredir.debian.org/debian/ bookworm-updates main non-free-
firmware
```

If you use Linux as a daily driver, you may end up with a repository list with many more entries, like on this Ubuntu system:

```
student@ubuntu:~$ wc -l /etc/apt/sources.list
63 /etc/apt/sources.list
```

There is much more to learn about `apt`, explore commands like `add-apt-repository`, `apt-key` and `apropos apt`.

16.3. the Red Hat package manager (rpm)

On Red Hat and other distros of that family, the *Red Hat package manager* (RPM) is used to install, upgrade and remove software. There's a basic command, `rpm`, and a more advanced tool, `dnf` (comparable with the situation on Debian-based systems, where `dpkg` is the basic tool and `apt` the more advanced one). When you install a graphical desktop, there's also a GUI tool for package management, but we won't be discussing that here.

Software distributed in the `rpm` format will have a file name following this format: `package-version-release.architecture.rpm`. For example, the package name `openssh-server-8.7p1-34.el9.x86_64.rpm` has the following components:

- package name: `openssh-server`
- version: `8.7p1`
- release: `34.el9` (el9 stands for Enterprise Linux 9, indicating it is compatible with RHEL 9)
- architecture: `x86_64` (suitable for a 64-bit Intel/AMD processor)

We will start with discussing the `dnf` command, since that one is most commonly used. After that, we'll show how to use the `rpm` command.

16.3.1. dnf

The name of the `dnf` command has a bit of a convoluted history. It stands for "Dandified Yum", and is a fork/improvement of the `yum` package manager command. Yum stands for *Yellowdog Updater, Modified*, and was originally developed for the now defunct Yellow Dog Linux distribution (for the IBM POWER7 processor). Red Hat started using it in RHEL 5 and it was the default package manager for Red Hat and its derivatives for many years. However, more recently, they developed `dnf` to replace `yum` with the former now being the default package manager for Fedora, Red Hat Enterprise Linux and its derivatives.

The `dnf` command works quite similarly to the `apt` command on Debian-based systems. It has similar subcommands, which we will discuss in the next sections. However, an equivalent for `apt update` does *not* exist. The `dnf` command will automatically update its package database whenever you execute it.

16.3.2. dnf list

Issue `dnf list` to see a list of all packages that DNF knows about.

```
[student@el ~]$ dnf list | wc -l
6751
[student@el ~]$ dnf list --all | wc -l
6751
```

Add the option `--available` or `--installed` to see only the packages that are available for installation or installed on the system.

```
[student@el ~]$ dnf list --available | wc -l
6392
[student@el ~]$ dnf list --installed | wc -l
353
```

Issue `dnf list $package` to get all versions (in different repositories) of one package.

```
[student@el ~]$ dnf list kernel
Last metadata expiration check: 0:12:15 ago on Sun 25 Feb 2024 07:16:59 PM UTC.
Installed Packages
kernel.x86_64              5.14.0-362.8.1.el9_3        @anaconda
kernel.x86_64              5.14.0-362.13.1.el9_3       @baseos
Available Packages
kernel.x86_64              5.14.0-362.18.1.el9_3       baseos
```

16.3.3. dnf search

To search for a package containing a certain string in the description or name use `dnf search $string`.

```
[student@el ~]$ dnf search openssh
Last metadata expiration check: 0:15:35 ago on Sun 25 Feb 2024 07:16:59 PM UTC.
========================= Name Exactly Matched: openssh ========================
openssh.x86_64 : An open source implementation of SSH protocol version 2
======================== Name & Summary Matched: openssh =======================
openssh-askpass.x86_64 : A passphrase dialog for OpenSSH and X
openssh-keycat.x86_64 : A mls keycat backend for openssh
============================ Name Matched: openssh =============================
```

```
openssh-clients.x86_64 : An open source SSH client applications
openssh-server.x86_64 : An open source SSH server daemon
[student@el ~]$ dnf search epel
Last metadata expiration check: 0:18:51 ago on Sun 25 Feb 2024 07:16:59 PM UTC.
============================== Name Matched: epel ==============================
epel-release.noarch : Extra Packages for Enterprise Linux repository configuration
```

16.3.4. dnf info

Information about a specific package can be obtained with `dnf info $package`.

```
[student@el ~]$ dnf info epel-release
Last metadata expiration check: 1:15:53 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
Installed Packages
Name         : epel-release
Version      : 9
Release      : 7.el9
Architecture : noarch
Size         : 26 k
Source       : epel-release-9-7.el9.src.rpm
Repository   : @System
From repo    : epel
Summary      : Extra Packages for Enterprise Linux repository configuration
URL          : http://download.fedoraproject.org/pub/epel
License      : GPLv2
Description  : This package contains the Extra Packages for Enterprise Linux
             : (EPEL) repository GPG key as well as configuration for yum.
```

This gives you a lot of information about the package, including the version, release, architecture, size, source, repository, summary, link to the project website, license and description.

If the repository is indicated as `@System`, it means that the package is installed. Otherwise, it would show the name of the repository from which the package would be installed.

```
[student@el ~]$ dnf info zork
Last metadata expiration check: 1:19:14 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
Available Packages
Name         : zork
Version      : 1.0.3
Release      : 5.el9
Architecture : x86_64
Size         : 179 k
Source       : zork-1.0.3-5.el9.src.rpm
Repository   : epel
Summary      : Public Domain original DUNGEON game (Zork I)
URL          : https://github.com/devshane/zork
License      : Public Domain
Description  : Public Domain source code to the original DUNGEON game (Zork I).
[ ... ]
```

16.3.5. dnf install

To install an application, use `dnf install $package`. Naturally, `dnf` will install all the necessary dependencies.

```
[student@el ~]$ sudo dnf install epel-release
Last metadata expiration check: 2:07:04 ago on Sun 25 Feb 2024 05:32:50 PM UTC.
Dependencies resolved.
================================================================================
 Package              Architecture    Version           Repository        Size
================================================================================
Installing:
 epel-release         noarch          9-5.el9           extras            18 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 18 k
Installed size: 25 k
Is this ok [y/N]: y
Downloading Packages:
epel-release-9-5.el9.noarch.rpm                     62 kB/s |  18 kB     00:00
--------------------------------------------------------------------------------
----
Total                                              23 kB/s |  18 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : epel-release-9-5.el9.noarch                            1/1
  Running scriptlet: epel-release-9-5.el9.noarch                            1/1
Many EPEL packages require the CodeReady Builder (CRB) repository.
It is recommended that you run /usr/bin/crb enable to enable the CRB repository.

  Verifying        : epel-release-9-5.el9.noarch                            1/1

Installed:
  epel-release-9-5.el9.noarch

Complete!
```

Add the option `-y` to skip confirmation. If the package is already installed, `install` will upgrade the package to the latest version.

```
[student@el ~]$ sudo dnf install -y sudo
Last metadata expiration check: 0:01:45 ago on Sun 25 Feb 2024 07:43:07 PM UTC.
Package sudo-1.9.5p2-9.el9.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package       Architecture    Version                   Repository       Size
================================================================================
Upgrading:
 sudo          x86_64          1.9.5p2-10.el9_3          baseos          1.0 M

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 1.0 M
Downloading Packages:
```

```
sudo-1.9.5p2-10.el9_3.x86_64.rpm                3.0 MB/s | 1.0 MB     00:00
--------------------------------------------------------------------------------
----
Total                                           1.3 MB/s | 1.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Upgrading        : sudo-1.9.5p2-10.el9_3.x86_64                           1/2
  Running scriptlet: sudo-1.9.5p2-10.el9_3.x86_64                           1/2
  Cleanup          : sudo-1.9.5p2-9.el9.x86_64                              2/2
  Running scriptlet: sudo-1.9.5p2-9.el9.x86_64                              2/2
  Verifying        : sudo-1.9.5p2-10.el9_3.x86_64                           1/2
  Verifying        : sudo-1.9.5p2-9.el9.x86_64                              2/2

Upgraded:
  sudo-1.9.5p2-10.el9_3.x86_64

Complete!
```

You can add more than one parameter here.

```
[student@el ~]$ sudo dnf install httpd mod_ssl mariadb-server php php-mysqlnd
```

16.3.6. dnf upgrade

To bring all applications up to date by downloading and installing them, issue `dnf upgrade`. All software that was installed via `dnf` will be updated to the latest version that is available in the repository.

```
[student@el ~]$ sudo dnf upgrade
Last metadata expiration check: 0:05:19 ago on Sun 25 Feb 2024 07:43:07 PM UTC.
Dependencies resolved.
================================================================================
 Package                 Arch    Version                     Repository  Size
================================================================================
Installing:
 kernel                  x86_64  5.14.0-362.18.1.el9_3       baseos     9.4 k
Upgrading:
 epel-release             noarch  9-7.el9                    epel        19 k
 gnutls                  x86_64  3.7.6-23.el9_3.3            baseos     1.0 M
[ ... ]

Transaction Summary
================================================================================
Install  10 Packages
Upgrade  12 Packages

Total download size: 89 M
Is this ok [y/N]: y
Downloading Packages:
(1/22): graphite2-1.3.14-9.el9.x86_64.rpm       189 kB/s |  94 kB     00:00
(2/22): freetype-2.10.4-9.el9.x86_64.rpm        752 kB/s | 387 kB     00:00
[ ... ]
Complete!
```

If you only want to update one package, use `dnf upgrade $package`. It behaves the same as `dnf install $package`.

16.3.7. dnf provides

To search for a package containing a certain file use `dnf provides $filename` (or globbing pattern). This is especially useful if you want to install a specific command that has a different name than the package name. For example, say that you've heard about the `ag` command that is a faster alternative to `grep`. The command `dnf search ag` spews out too much output, so no useful results:

```
[student@el ~]$ dnf search ag | wc -l
Last metadata expiration check: 0:02:48 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
2979
```

Listing available packages with `ag` shows that there is no such package:

```
[student@el ~]$ dnf list --available ag
Last metadata expiration check: 0:04:05 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
Error: No matching Packages to list
[student@el ~]$ dnf list --available ag*
Last metadata expiration check: 0:04:09 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
Available Packages
Agda.x86_64                   2.6.2.2-36.el9                              epel
Agda-common.noarch            2.6.2.2-36.el9                              epel
aggregate6.noarch             1.0.12-2.el9                                epel
agrep.x86_64                 0.8.0-34.20140228gitc2f5d13.el9              epel
```

The last package looks promising, but it's not the one we're looking for. So let's use `dnf provides` to find out which package contains the `ag` command. If the command is `ag`, we expect that it is installed in one of the `bin/` directories, i.e. `/bin`, `/usr/bin`, `/sbin`, `/usr/sbin`, `/usr/local/bin`, `/usr/local/sbin`, `/usr/local/bin`, `/usr/local/sbin`. We can summarize the possible path names with globbing pattern `*bin/ag`:

```
[student@el ~]$ dnf provides *bin/ag
Last metadata expiration check: 0:07:13 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
the_silver_searcher-2.2.0^2020704.5a1c8d8-3.el9.x86_64 : Super-fast text
                                                       : searching tool (ag)
Repo        : epel
Matched from:
Other       : *bin/ag
[student@el ~]$ sudo dnf install -y the_silver_searcher
```

So the name of the package is `the_silver_searcher` (`ag` being the chemical symbol for silver) and it is provided by the EPEL repository (Extra Packages for Enterprise Linux). We can install it with `dnf install the_silver_searcher`.

16.3.8. dnf remove

Removing a package is done with `dnf remove $package`. This will remove the package and all its dependencies that are not needed by other packages.

```
[student@el ~]$ sudo dnf remove net-tools
Dependencies resolved.
================================================================================
 Package        Arch        Version                        Repository      Size
================================================================================
Removing:
 net-tools      x86_64      2.0-0.62.20160912git.el9       @anaconda      912 k

Transaction Summary
================================================================================
Remove  1 Package

Freed space: 912 k
Is this ok [y/N]: y
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Erasing          : net-tools-2.0-0.62.20160912git.el9.x86_64              1/1
  Verifying        : net-tools-2.0-0.62.20160912git.el9.x86_64              1/1

Removed:
  net-tools-2.0-0.62.20160912git.el9.x86_64

Complete!
```

By the way, this package, `net-tools`, contains commands that are considered to be obsolete and have been replaced by other, newer implementations. You don't really need it, so it's a good example for this section. If you removed it, feel free to reinstall it if you want!

16.3.9. dnf software groups

Issue `dnf grouplist` to see a list of all available software groups.

```
[student@el ~]$ dnf grouplist
Last metadata expiration check: 1:00:37 ago on Sun 25 Feb 2024 07:55:24 PM UTC.
Available Environment Groups:
   Server with GUI
   Server
   Minimal Install
   Workstation
   KDE Plasma Workspaces
   Virtualization Host
   Custom Operating System
Available Groups:
   RPM Development Tools
   .NET Development
   Container Management
   Console Internet Tools
   Graphical Administration Tools
   Scientific Support
   Headless Management
   Smart Card Support
   Legacy UNIX Compatibility
   Security Tools
```

```
   Network Servers
   System Tools
   Development Tools
   Fedora Packager
   VideoLAN Client
   Xfce
```

To install a set of applications, brought together via a group, use `yum groupinstall $group-name`.

```
[student@el ~]$ sudo dnf groupinstall 'Security Tools'
Last metadata expiration check: 1:00:35 ago on Sun 25 Feb 2024 08:03:34 PM UTC.
Dependencies resolved.
================================================================================
 Package               Arch      Version                    Repository    Size
================================================================================
Installing group/module packages:
 scap-security-guide    noarch    0.1.69-3.el9_3.alma.1     appstream    813 k
Installing dependencies:
 libtool-ltdl          x86_64    2.4.6-45.el9              appstream     36 k
 libxslt               x86_64    1.1.34-9.el9              appstream    240 k
 openscap              x86_64    1:1.3.8-1.el9_2.alma.2    appstream    1.9 M
 openscap-scanner      x86_64    1:1.3.8-1.el9_2.alma.2    appstream     57 k
 xml-common            noarch    0.6.3-58.el9              appstream     31 k
 xmlsec1               x86_64    1.2.29-9.el9              appstream    189 k
 xmlsec1-openssl       x86_64    1.2.29-9.el9              appstream     90 k
Installing Groups:
 Security Tools

Transaction Summary
================================================================================
Install  8 Packages

Total download size: 3.3 M
Installed size: 103 M
Is this ok [y/N]:
[ ... ]
```

Read the manual page of `dnf` for more information about managing groups in `dnf`. In practice, chances are that you won't need this feature very often.

16.3.10. rpm -qa

In the following sections, we'll show what you can do with the `rpm` command.

To obtain a list of all installed software, use the `rpm -qa` command.

```
[student@el ~]$ rpm -qa | grep ssh
libssh-config-0.10.4-11.el9.noarch
libssh-0.10.4-11.el9.x86_64
openssh-8.7p1-34.el9.x86_64
openssh-clients-8.7p1-34.el9.x86_64
openssh-server-8.7p1-34.el9.x86_64
```

16.3.11. rpm -q

To verify whether one package is installed, use `rpm -q`.

```
[student@el ~]$ rpm -q vim-enhanced
package vim-enhanced is not installed
[student@el ~]$ rpm -q vim-minimal
vim-minimal-8.2.2637-20.el9_1.x86_64
[student@el ~]$ rpm -q kernel
kernel-5.14.0-362.8.1.el9_3.x86_64
kernel-5.14.0-362.13.1.el9_3.x86_64
kernel-5.14.0-362.18.1.el9_3.x86_64
```

16.3.12. rpm -ql

To see which files are installed by a package, use `rpm -ql`.

```
[student@el ~]$ rpm -ql vim-minimal
/etc/virc
/usr/bin/ex
/usr/bin/rvi
/usr/bin/rview
/usr/bin/vi
/usr/bin/view
/usr/lib/.build-id
/usr/lib/.build-id/c6
/usr/lib/.build-id/c6/aa3d8d79f09dd48e99475c332bed4df39d76e1
/usr/libexec/vi
/usr/share/man/man1/ex.1.gz
/usr/share/man/man1/rvi.1.gz
/usr/share/man/man1/rview.1.gz
/usr/share/man/man1/vi.1.gz
/usr/share/man/man1/view.1.gz
/usr/share/man/man5/virc.5.gz
```

16.3.13. rpm -Uvh

To install or upgrade a package, use the -Uvh switches. The -U switch is the same as -i for install, except that older versions of the software are removed. The -vh switches are for nicer output.

You would typically use this command to install an .rpm package that you have downloaded from the internet. Beware, though, that `rpm` does not resolve dependencies, so you might need to install other packages first.

```
[student@el ~]$ sudo rpm -Uvh ./htop-3.3.0-1.el9.x86_64.rpm
error: Failed dependencies:
        libhwloc.so.15()(64bit) is needed by htop-3.3.0-1.el9.x86_64
```

16.3.14. rpm -e

To remove a package, use the -e switch.

```
[student@el ~]$ rpm -q net-tools
net-tools-2.0-0.62.20160912git.el9.x86_64
[student@el ~]$ sudo rpm -e net-tools
[student@el ~]$ rpm -q net-tools
package net-tools is not installed
```

`rpm -e` verifies dependencies, and thus will prevent you from accidentailly erasing packages that are needed by other packages.

```
[student@el ~]$ sudo rpm -e slang
error: Failed dependencies:
            libslang.so.2()(64bit) is needed by (installed) newt-0.52.21-
11.el9.x86_64
        libslang.so.2(SLANG2)(64bit) is needed by (installed) newt-0.52.21-
11.el9.x86_64
```

16.3.15. Package cache

When `dnf` installs or upgrades a package, it will download the package from the repository and store it temporarily in the cache. The cache also contains repository metadata. The default location of the cache is `/var/cache/dnf`. You can clean the cache with `dnf clean all`.

```
[student@el ~]$ dnf clean all
51 files removed
```

Remark that .rpm files will normally be removed automatically after they were installed successfully. You can change this behavior in `/etc/dnf/dnf.conf` by setting `keepcache=1`.

16.3.16. Configuration

The main configuration file for `dnf` is `/etc/dnf/dnf.conf`. This file contains a few basic settings. The location of package repositories that are available to the system are kept in the directory `/etc/yum.repos.d/`. Each repository has its own file, with a `.repo` extension.

```
[student@el ~]$ ls /etc/yum.repos.d/
almalinux-appstream.repo         almalinux-resilientstorage.repo
almalinux-baseos.repo            almalinux-rt.repo
almalinux-crb.repo               almalinux-saphana.repo
almalinux-extras.repo            almalinux-sap.repo
almalinux-highavailability.repo  epel-cisco-openh264.repo
almalinux-nfv.repo               epel.repo
almalinux-plus.repo              epel-testing.repo
```

A repo file is a text file in the INI format, and contains information about the repository, such as the name, the base URL, the GPG key, etc. Here's an example with part of the contents of the `epel.repo` file:

```
[epel]
name=Extra Packages for Enterprise Linux $releasever - $basearch
## It is much more secure to use the metalink, but if you wish to use a local
 ↪  mirror
## place its address here.
#baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/
 ↪
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-
 ↪  $releasever&arch=$basearch&infra=$infra&content=$contentdir
enabled=1
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever
```

16.3.17. Working with multiple repositories

You can get a list of the currently enabled repositories with `dnf repolist`.

```
[student@el ~]$ dnf repolist
repo id              repo name
appstream            AlmaLinux 9 - AppStream
baseos               AlmaLinux 9 - BaseOS
epel                 Extra Packages for Enterprise Linux 9 - x86_64
epel-cisco-openh264 Extra Packages for Enterprise Linux 9 openh264 (From Cisco) -
x86_64
extras               AlmaLinux 9 - Extras
```

And specific information about a repository with `dnf repoinfo $repo`.

```
[student@el ~]$ dnf repoinfo epel-testing
Last metadata expiration check: 0:02:40 ago on Sun 25 Feb 2024 10:32:04 PM UTC.
Repo-id            : epel-testing
Repo-name          : Extra Packages for Enterprise Linux 9 - Testing - x86_64
Repo-status        : disabled
Repo-metalink      : https://mirrors.fedoraproject.org/metalink?repo=testing-
epel9&arch=x86_64&infra=$infra&content=$contentdir
Repo-expire        : 172,800 second(s) (last: unknown)
Repo-filename      : /etc/yum.repos.d/epel-testing.repo
Total packages: 0
```

One important flag for dnf is `--enablerepo`. Use this command if you want to use a repository that is not enabled by default. For example, let's say you want to install the latest version of `fail2ban`, but the one in the "normal" repository is too old:

```
[student@el ~]$ dnf list --available fail2ban
Last metadata expiration check: 0:01:44 ago on Sun 25 Feb 2024 10:32:04 PM UTC.
Available Packages
fail2ban.noarch                        1.0.2-7.el9                        epel
```

Maybe `epel-testing` has a newer version:

```
[student@el ~]$ dnf list --available --repo epel-testing fail2ban
Last metadata expiration check: 0:06:10 ago on Sun 25 Feb 2024 10:30:33 PM UTC.
Available Packages
fail2ban.noarch                    1.0.2-12.el9                    epel-testing
```

It does, but you won't be able to install it due to the fact that `epel-testing` is disabled. However, you can temporarily enable it with the `--enablerepo` flag:

```
[student@el ~]$ sudo dnf install --enablerepo=epel-testing fail2ban
[sudo] password for student:
Last metadata expiration check: 0:13:34 ago on Sun 25 Feb 2024 10:24:12 PM UTC.
Dependencies resolved.
================================================================================
 Package                Arch       Version            Repository         Size
================================================================================
Installing:
 fail2ban               noarch     1.0.2-12.el9       epel-testing      8.8 k
Installing dependencies:
 esmtp                  x86_64      1.2-19.el9         epel              52 k
 fail2ban-firewalld     noarch      1.0.2-12.el9       epel-testing     8.9 k
 fail2ban-selinux       noarch      1.0.2-12.el9       epel-testing      29 k
 fail2ban-sendmail      noarch      1.0.2-12.el9       epel-testing      12 k
 fail2ban-server        noarch      1.0.2-12.el9       epel-testing     444 k
 libesmtp               x86_64      1.0.6-24.el9       epel              66 k
 liblockfile            x86_64      1.14-10.el9        baseos            28 k

Transaction Summary
================================================================================
Install  8 Packages

Total download size: 647 k
Installed size: 1.8 M
Is this ok [y/N]:
```

16.4. pip, the Python package manager

Some programming languages, a.o. Python, have their own package management system that allows you to install applications and/or libraries. In the case of Python, the package manager is called `pip`. It is used to install Python packages from the Python Package Index (PyPI). In fact, there are multiple package managers for Python (a.o. easy_install, conda, etc.), but `pip` is the most widely used.

As a system administrator, or as an end user, this sometimes puts you in a difficult position. Some widely known and used Python libraries can be installed both through your distribution's package manager, and through pip. Which one to choose is not always clear. In general, it is best to use the distribution's package manager, as it will integrate the package into the system and will be updated when the system is updated. However, some packages are not available in the distribution's repositories, or the version you get with `pip` is more recent. In that case, you can use `pip` to install the package.

Another thing to note is that `pip` can be used as a normal user, or as root, and in each case it will install the package in a different location. When you install a package as a normal user, it will be installed in your home directory, and will only be available to you. When you install a package as root, it will be installed system-wide, and will be available to all users. However, if you install a package as root, you will get a warning message:

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting beha
```

A virtual environment is a way to create an isolated environment for a Python project, where you can install packages without affecting the system's Python installation. This is especially useful when you are developing Python applications, and you want to make sure that the

libraries you use are the same as the ones used in production. Using and managing virtual environments is beyond the scope of this course, but you can find more information in the Python documentation.

As general guidelines, we suggest the following:

- If the library or application is available in the distribution's repositories, use the distribution's package manager to install it.
- Avoid installing Python libraries or applications system-wide as root using `pip`.
- Normal users may use `pip` to install Python libraries or applications in their home directory.

16.4.1. installing pip

`pip` may not be installed by default on your system. You can install it using your distribution's package manager. For example, on Debian-based systems, you can install it using `apt`:

```
student@debian:~$ sudo apt install python3-pip
```

On Red Hat-based systems, you can install it using `dnf`:

```
student@el ~$ sudo dnf install python3-pip
```

16.4.2. listing packages

You can list the packages installed with `pip` using the `list` command:

```
student@linux:~$ pip list
Package         Version
--------------- --------
dbus-python     1.2.18
distro          1.5.0
gpg             1.15.1
libcomps        0.1.18
nftables        0.1
pip             21.2.3
PyGObject       3.40.1
python-dateutil 2.8.1
PyYAML          5.4.1
rpm             4.16.1.3
selinux         3.5
sepolicy        3.5
setools         4.4.3
setuptools      53.0.0
six             1.15.0
systemd-python  234
```

16.4.3. searching for packages

Searching for packages can **NOT** be done on the command line. To search for packages, you can use the Python Package Index website instead. If you try `pip search`, you will get an error message:

```
student@linux:~$ pip search ansible
ERROR: XMLRPC request failed [code: -32500]
RuntimeError: PyPI no longer supports 'pip search' (or XML-RPC search). Please use https://p
reference/xml-rpc.html#deprecated-methods for more information.
```

16.4.4. installing packages

You can install a package using the `install` command:

```
student@linux:~$ pip install ansible
```

Just like `apt` and `dnf`, `pip` will install the package and its dependencies.

16.4.5. removing packages

Uninstalling a package is done with the `uninstall` command:

```
student@linux:~$ pip uninstall ansible
```

Unfortunately, dependencies are not removed when you uninstall a package with `pip`.

16.5. container-based package managers

With the release of Docker, container-based virtualization has become very popular as a method of distributing and deploying applications on servers. One of the advantages of containers is that they offer a sandbox environment for applications, meaning the application and its dependencies are isolated from the rest of the system. This makes it possible to run applications with different dependencies on the same server, without the risk of conflicts. Containers are also very lightweight, they don't impose much overhead on the host system.

Now, there is no reason why containers can't be used to deploy applications on desktop systems as well. In fact, there are several container-based package managers that allow you to install and run applications in containers on your desktop. The advantage is that third party software vendors can distribute their applications independent of the Linux distribution, so they don't need to maintain different packages for (each family of) distribution(s). The disadvantage is that each application comes with their own dependencies, so you lose the advantage of sharing libraries between applications. Also, since the application is running in a container, it may not integrate well with the rest of the system, or may have only limited permissions to access files or other resources on your computer.

As with many Linux-based technologies, there are multiple tools to choose from. The most popular ones are Flatpak and Snap.

16.5.1. flatpak

Flatpak is a container-based package manager developed by an independent community of contributors, volunteers and supporting organizations. It is available for most Linux distributions and is supported by a large number of third party software vendors. Red Hat was one of the first to endorse Flatpak, and many others followed. Fedora Silverblue is a variant of Fedora that uses Flatpak as its primary package manager. Linux Mint also has Flatpak support enabled by default: in the Software Manager, some applications like Bitwarden, Slack, VS Code, etc. are available as Flatpaks.

If you want to use a container based package manager, Flatpak is probably the best choice for any Linux distribution other than Ubuntu.

In the following example, we'll install the open source password manager Bitwarden with Flatpak on a Linux Mint system. Remark that you don't need to be root to install Flatpak applications!

```
student@mint:~$ flatpak search Bitwarden
Name     Description                    Application ID      Version  Branch Remotes
Bitwarden  A secure and free password manager for com.bitwarden.desktop  2024.2.0 stable fla
Goldwarden A Bitwarden compatible desktop client  com.quexten.Goldwarden 0.2.13  stable fla
student@mint:~$ flatpak install Bitwarden
Looking for matches…
Found ref ‘app/com.bitwarden.desktop/x86_64/stable’ in remote ‘flathub’ (system).
Use this ref? [Y/n]: y
Required runtime for com.bitwarden.desktop/x86_64/stable (runtime/org.freedesktop.Platform
Do you want to install it? [Y/n]: y

com.bitwarden.desktop permissions:
   ipc              network               wayland     x11     dri     file access [1]
    dbus access [2]        system dbus access [3]

    [1] xdg-download
   [2] com.canonical.AppMenu.Registrar, org.freedesktop.Notifications, org.freedesktop.se
    [3] org.freedesktop.login1


        ID                                      Branch      Op Remote  Download
 1. [✓] com.bitwarden.desktop.Locale      stable    i  flathub 300.7 kB / 9.8 MB
 2. [✓] org.freedesktop.Platform.GL.default  23.08     i  flathub 162.0 MB / 162.3 MB
 3. [✓] org.freedesktop.Platform.GL.default  23.08-extra i  flathub  17.9 MB / 162.3 MB
 4. [✓] org.freedesktop.Platform.Locale    23.08     i  flathub  17.9 kB / 359.9 MB
 5. [✓] org.freedesktop.Platform         23.08     i  flathub 171.6 MB / 225.6 MB
 6. [✓] com.bitwarden.desktop           stable    i  flathub 132.5 MB / 133.4 MB

Installation complete.
```

To remove a Flatpak application, you can use the `uninstall` command:

```
student@mint:~$ flatpak uninstall Bitwarden
Found installed ref ‘app/com.bitwarden.desktop/x86_64/stable’ (system). Is this correct? [Y


        ID                                      Branch         Op
 1. [-] com.bitwarden.desktop                 stable         r
 2. [-] com.bitwarden.desktop.Locale          stable         r

Uninstall complete.
```

16.5.2. snap

Snap was developed by Canonical and is installed by default on Ubuntu. It is also available for other distributions (like the official Ubuntu derivatives, Solus and Zorin OS), but it is not as widely supported as Flatpak. Snap was also designed to work for cloud applications and Internet of Things devices.

In the following example, we'll install Grafana on an Ubuntu Server system.

```
student@ubuntu:~$ snap search grafana
Name          Version Publisher  Notes Summary
grafana     6.7.4  canonical✓ -    feature rich metrics dashboard and graph editor
grafana-agent 0.35.4  0×12b      -     Telemetry Agent
[ ... ]
student@ubuntu:~$ sudo snap install grafana
grafana 6.7.4 from Canonical✓ installed
```

To uninstall a Snap application, you can use the `remove` command:

```
student@ubuntu:~$ sudo snap remove grafana
grafana removed
```

16.6. downloading software outside the repository

These days, the case where you need software that is not available as a binary package has become exceedingly rare. However, *if* you want to install some experimental tool that hasn't been packaged yet, or you want to test the very latest experimental version of an application, you may have to download the source code and compile it yourself. Usually, the source code is available on the project's website or on a code hosting platform like GitHub, GitLab or Bitbucket. You then either download the source code as a `tgz`, `.tar.gz`, `.tar.bz2`, `tar.xz` file (also called a *tarball*) or you can clone the repository using `git`.

In the example below, we assume that you have downloaded the source code of an application written in C or C++, as is common for many Linux applications. Remark that in order to be able to compile the source code, you need to have the C compiler `gcc` and the build tool `make` installed on your system. You can install these using your distribution's package manager. Also, many applications depend on other libraries, which also have to be installed as source.

16.6.1. example: compiling zork

As an example, we will download the source code for Zork, an ancient text based adventure game, and compile it on a Fedora system. The source code is available on GitHub. We have installed `git`, `gcc` and `make` beforehand.

```
[student@fedora ~]$ git clone https://github.com/devshane/zork.git
Cloning into 'zork' ...
remote: Enumerating objects: 79, done.
remote: Total 79 (delta 0), reused 0 (delta 0), pack-reused 79
Receiving objects: 100% (79/79), 241.70 KiB | 2.14 MiB/s, done.
Resolving deltas: 100% (20/20), done.
[student@fedora ~]$ cd zork/
[student@fedora zork]$ ls
actors.c  demons.c  dmain.c  dso3.c  dso6.c  dtextc.dat  dverb2.c  history  Makefile  np2.c  n
ballop.c  dgame.c   dso1.c   dso4.c  dso7.c  dungeon.6   funcs.h   lightp.c  nobjs.c   np3.c  obj
clockr.c  dinit.c   dso2.c   dso5.c  dsub.c  dverb1.c    gdt.c     local.c   np1.c     np.c   parse
[student@fedora ~]$ make
cc -g    -c -o actors.o actors.c
cc -g    -c -o ballop.o ballop.c
cc -g    -c -o clockr.o clockr.c
[ ... etc ... ]
cc -g  -o zork actors.o ballop.o clockr.o demons.o dgame.o dinit.o dmain.o dso1.o dso2.o dso3
ltermcap
/usr/bin/ld: cannot find -ltermcap: No such file or directory
collect2: error: ld returned 1 exit status
make: *** [Makefile:69: dungeon] Error 1
```

As you can see, the `make` command fails because it cannot find the `termcap` library. This is a library that is used to control the terminal, and it is not installed on our system. This is a common problem when you try to install packages from source. You need to install these dependencies yourself and these are not always easy to find. In this case, we can install the `ncurses-devel` library, which is a modern replacement for `termcap`. How did we now that?

We used `dnf provides` to find library files that contain the string `termcap` (remark that the command took a long time to finish):

```
[student@fedora zork]$ dnf provides '*libtermcap.so*'
Last metadata expiration check: 1:56:05 ago on Mon 26 Feb 2024 05:46:43 PM UTC.
ncurses-devel-6.4-7.20230520.fc39.i686 : Development files for the ncurses library
Repo        : fedora
Matched from:
Other       : *libtermcap.so*
[student@fedora ~]$ sudo dnf install ncurses-devel
[ ... etc ... ]
```

Let's try to compile again:

```
[student@fedora zork]$ make
cc -g    -c -o actors.o actors.c
cc -g    -c -o ballop.o ballop.c
[ ... etc ... ]
cc -g    -c -o villns.o villns.c
cc -g  -o zork actors.o ballop.o clockr.o demons.o dgame.o dinit.o dmain.o dso1.o dso2.o dso3
ltermcap
[student@fedora zork]$
```

The command seems to have succeeded. The current directory now contains a new file called `zork`. This is the compiled application and it has execute permissions. You can run it by typing `./zork`:

```
[student@fedora zork]$ ls -l zork
-rwxr-xr-x. 1 vagrant vagrant 400968 Feb 26 19:45 zork
[student@fedora zork]$ file zork
zork: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /
linux-x86-64.so.2, BuildID[sha1]=3089e3cb1c1a7fc1cc1db41c3aa578c0b52f83f3, for GNU/Linux 3
[student@fedora zork]$ ./zork
Welcome to Dungeon.                   This version created 11-MAR-91.
You are in an open field west of a big white house with a boarded
front door.
There is a small mailbox here.
>
```

In this case, installing the game is as simple as copying the `zork` file to a directory in your `PATH`, like `/usr/local/bin` or (for a computer game) `/usr/local/games`. However, most Makefiles provide a way to install the application in the system, usually by running `make install`. This will copy the executable, manual pages and other documentation to the correct location.

```
[student@fedora zork]$ sudo make install
mkdir -p /usr/games  /usr/share/man/man6
cp zork /usr/games
cp dtextc.dat /usr/games/lib
cp dungeon.6 /usr/share/man/man6/
```

Remark that the "official" location where manually installed applications belong in a Linux directory structure is `/usr/local` (for applications that follow the Filesystem Hierarchy Standard) or `/opt` (for applications that want to keep all files in a single directory).

16.6.2. installing from a tarball

Before unpacking a tarball, it's useful to check its contents:

```
student@linux:~$ tar tf $downloadedFile.tgz
```

The `t` option lists the content of the archive, `f` should be followed by the filename of the tarball. For `.tgz`, you may add option `z` and for `.tar.bz2` option `j`. However, the `tar` command should recognize the compression method automatically.

Check whether the package archive unpacks in a subdirectory (which is the preferred case) or in the current directory and create a subdirectory yourself if necessary. After that, you can unpack the tarball:

```
student@linux:~$ tar xf $downloadedFile.tgz
```

Now, be sure to read the README file carefully! Normally the readme will explain what to do after download.

Usually the steps are always the same three:

1. running a script `./configure`. It will gather information about your system that is needed to compile the software so that it can actually run on your system
2. executing the command `make` (which is the actual compiling)
3. finally, executing `make install` to copy the files to their proper location.

16.7. practice: package management

1. Verify whether `gcc`, `sudo` and `zork` are installed.
2. Use dnf or apt to search for and install the `scp`, `tmux`, and `man`-pages packages. Did you find them all?
3. Search the internet for 'webmin' and figure out how to install it.
4. If time permits, search for and install samba including the samba docs pdf files (thousands of pages in two pdf's).

16.8. solution: package management

1. Verify whether gcc, sudo and zork are installed.

   On Enterprise Linux:

   ```
   rpm -qa | grep gcc
   rpm -qa | grep sudo
   rpm -qa | grep zork
   ```

   On Debian/Ubuntu:

   ```
   dpkg -l | grep gcc
   dpkg -l | grep sudo
   dpkg -l | grep zork
   ```

2. Use dnf or apt to search for and install the `scp`, `tmux`, and `man-pages` packages. Did you find them all ?

   On Red Hat/CentOS:

```
dnf search scp
dnf search tmux
dnf search man-pages
```

On Debian/Ubuntu:

```
apt search scp
apt search tmux
apt search man-pages
```

3. Search the internet for 'webmin' and figure out how to install it.

   Google should point you to webmin.com. The download page helps you to download a repository file so you can install webmin with your package manager. The latest Webmin distribution is available in various package formats for download, a.o. .rpm, .deb, etc.

4. If time permits, search for and install samba including the samba docs pdf files (thousands of pages in two pdf's).

17. general networking

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

While this chapter is not directly about `Linux`, it does contain general networking concepts that will help you in troubleshooting networks on `Linux`.

17.1. network layers

17.1.1. seven OSI layers

When talking about protocol layers, people usually mention the seven layers of the `osi` protocol (Application, Presentation, Session, Transport, Network, Data Link and Physical). We will discuss layers 2 and 3 in depth, and focus less on the other layers. The reason is that these layers are important for understanding networks. You will hear administrators use words like "this is a layer 2 device" or "this is a layer 3 broadcast", and you should be able to understand what they are talking about.

17.1.2. four DoD layers

The `DoD` (or tcp/ip) model has only four layers, roughly mapping its `network access layer` to OSI layers 1 and 2 (Physical and Datalink), its `internet` (IP) layer to the OSI `network layer`, its `host-to-host` (tcp, udp) layer to OSI layer 4 (transport) and its `application layer` to OSI layers 5, 6 and 7.

Below an attempt to put OSI and DoD layers next to some protocols and devices.

<table>
<tr><th>OSI Model</th><th>DoD Model</th><th colspan="2">protocols</th><th>devices/apps</th></tr>
<tr><td>layer 5, 6, 7</td><td>application</td><td colspan="2">dns, dhcp, ntp, snmp, https, ftp, ssh, telnet, http, pop3... others</td><td>web server, mail server, browser, mail client...</td></tr>
<tr><td>layer 4</td><td>host-to-host</td><td>tcp</td><td>udp</td><td>gateway</td></tr>
<tr><td>layer 3</td><td>internet</td><td colspan="2">ip, icmp, igmp</td><td>router, firewall layer 3 switch</td></tr>
<tr><td>layer 2</td><td rowspan="2">network access</td><td colspan="2">arp (mac), rarp</td><td>bridge layer 2 switch</td></tr>
<tr><td>layer 1</td><td colspan="2">ethernet, token ring</td><td>hub</td></tr>
</table>

17.1.3. short introduction to the physical layer

The physical layer, or `layer 1`, is all about voltage, electrical signals and mechanical connections. Some networks might still use `coax` cables, but most will have migrated to `utp` (cat 5 or better) with `rj45` connectors.

Devices like `repeaters` and `hubs` are part of this layer. You cannot use software to 'see' a `repeater` or `hub` on the network. The only thing these devices are doing is amplifying electrical signals on cables. `Passive hubs` are multiport amplifiers that amplify an incoming electrical signal on all other connections. `Active hubs` do this by reading and retransmitting bits, without interpreting any meaning in those bits.

Network technologies like `csma/cd` and `token ring` are defined on this layer.

This is all we have to say about `layer 1` in this book.

17.1.4. short introduction to the data link layer

The data link layer, or `layer 2` is about frames. A frame has a `crc` (cyclic redundancy check). In the case of ethernet (802.3), each network card is identifiable by a unique 48-bit `mac` address (media access control address).

On this layer we find devices like bridges and switches. A bridge is more intelligent than a hub because a `bridge` can make decisions based on the mac address of computers. A `switch` also understands mac addresses.

In this book we will discuss commands like `arp` and `ifconfig` to explore this layer.

17.1.5. short introduction to the network layer

`Layer 3` is about ip packets. This layer gives every host a unique 32-bit ip address. But `ip` is not the only protocol on this layer, there is also icmp, igmp, ipv6 and more. A complete list can be found in the `/etc/protocols` file.

On this layer we find devices like `routers` and layer 3 switches, devices that know (and have) an ip address.

In tcp/ip this layer is commonly referred to as the `internet layer`.

17.1.6. short introduction to the transport layer

We will discuss the `tcp` and `udp` protocols in the context of layer 4. The DoD model calls this the host-to-host layer.

17.1.7. layers 5, 6 and 7

The tcp/ip application layer includes layers 5, 6 and 7. Details on the difference between these layers are out of scope of this course.

17.1.8. network layers in this book

Stacking of layers in this book is based on the `Protocols in Frame` explanation in the `wireshark` sniffer. When sniffing a dhcp packet, we notice the following in the sniffer.

```
[Protocols in Frame: eth:ip:udp:bootp]
```

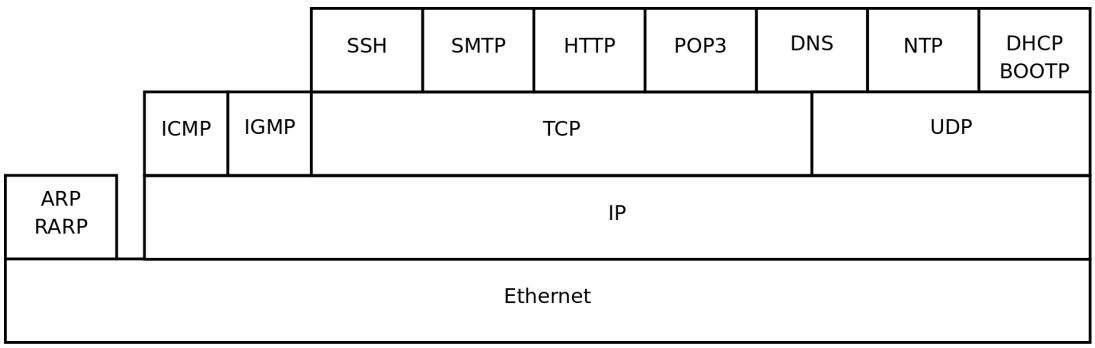
Sniffing for ntp (Network Time Protocol) packets gives us this line, which makes us conclude to put `ntp` next to `bootp` in the protocol chart below.

```
[Protocols in Frame: eth:ip:udp:ntp]
```

Sniffing an `arp` broadcast makes us put arp next to `ip`. All these protocols are explained later in this chapter.

```
[Protocols in Frame: eth:arp]
```

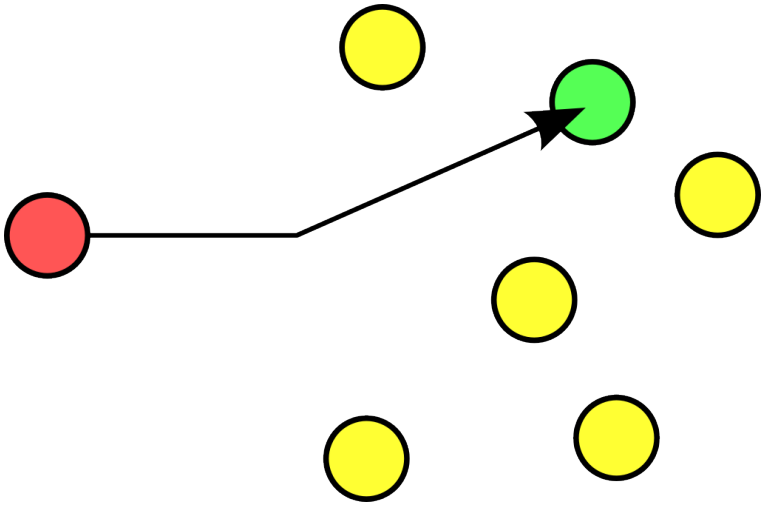
Below is a protocol chart based on wireshark's knowledge. It contains some very common protocols that are discussed in this book. The chart does not contain all protocols.



17.2. unicast, multicast, broadcast, anycast

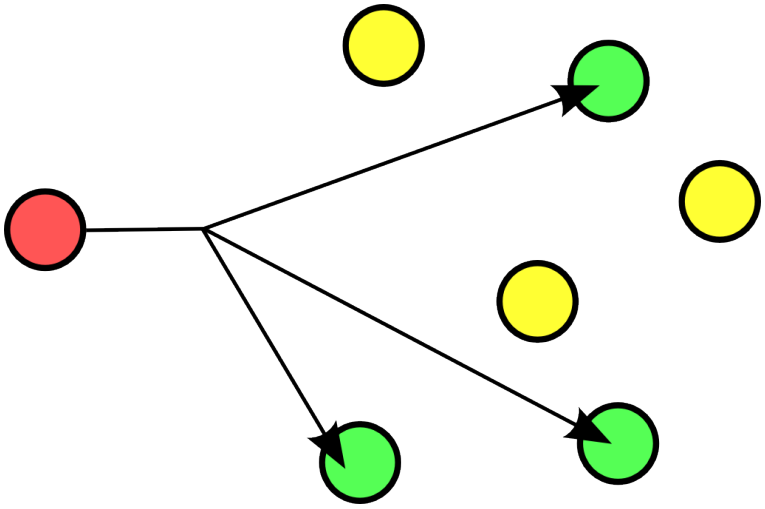
17.2.1. unicast

A `unicast` communication originates from one computer and is destined for exactly one other computer (or host). It is common for computers to have many `unicast` communications.



17.2.2. multicast

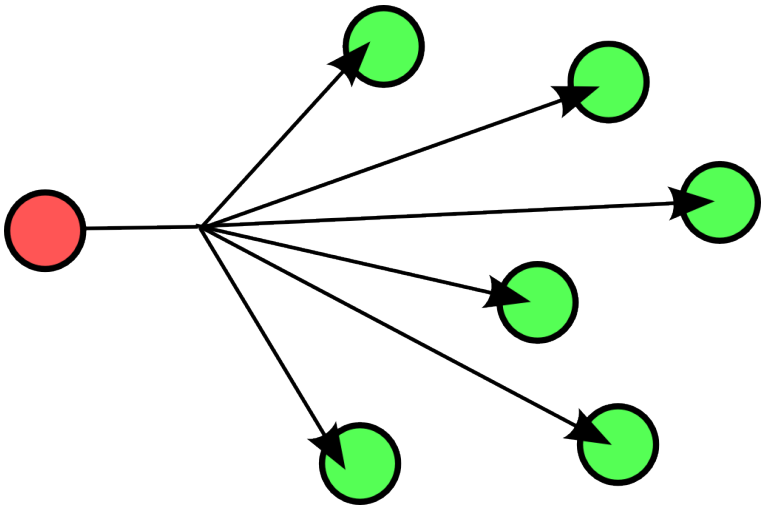
A `multicast` is destined for a group (of computers).



Some examples of `multicast` are Realplayer (.sdp files) and `ripv2` (a routing protocol).

17.2.3. broadcast

A `broadcast` is meant for everyone.

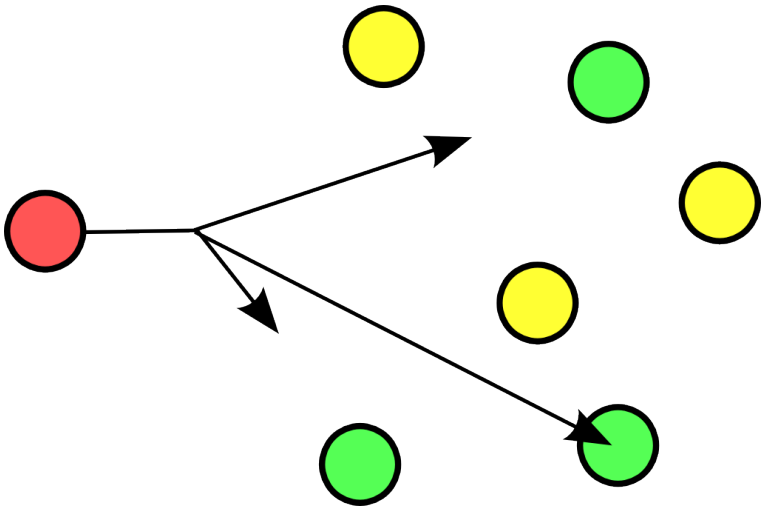


Typical example here is the BBC (British Broadcasting Corporation) broadcasting to everyone. In datacommunications a broadcast is most common confined to the `lan`.

Careful, a `layer 2 broadcast` is very different from a `layer 3 broadcast`. A layer two broadcast is received by all network cards on the same segment (it does not pass any router), whereas a layer 3 broadcast is received by all hosts in the same ip subnet.

17.2.4. anycast

The `root name servers` of the internet use `anycast`. An `anycast` signal goes the the (geographically) nearest of a well defined group.



With thanks to the nice anonymous wikipedia contributor to put these pictures in the public domain.

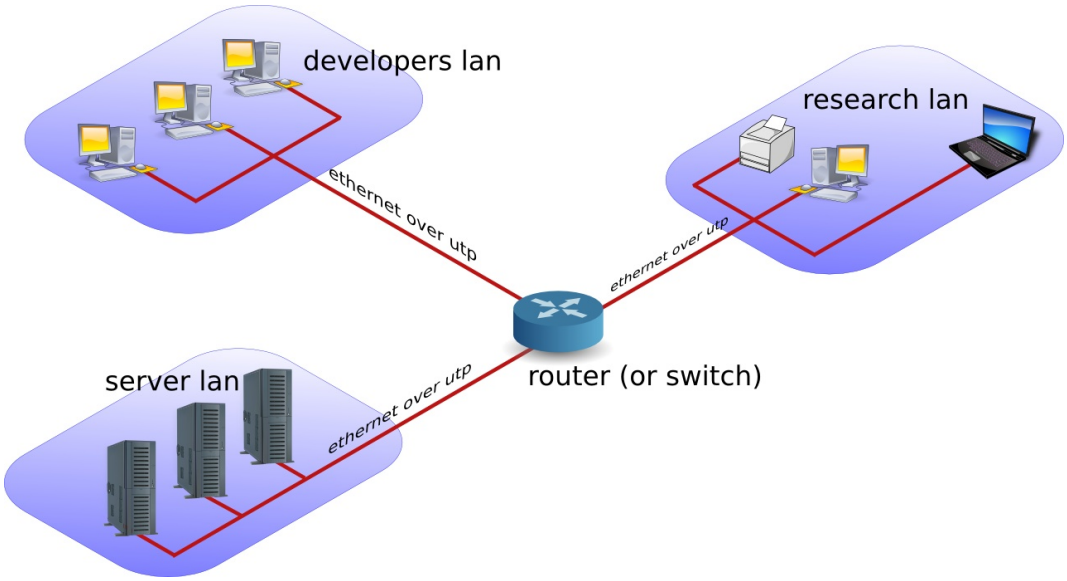
17.3. lan-wan-man

The term `lan` is used for local area networks, as opposed to a `wan` for wide area networks. The difference between the two is determined by the `distance` between the computers, and not by the number of computers in a network. Some protocols like `atm` are designed for use in a `wan`, others like `ethernet` are designed for use in a `lan`.

17.3.1. lan

A `lan` (Local Area Network) is a local network. This can be one room, or one floor, or even one big building. We say `lan` as long as computers are `close` to each other. You can also define a `lan` when all computers are `ethernet` connected.

A `lan` can contain multiple smaller `lan`'s. The picture below shows three `lan`'s that together make up one `lan`.



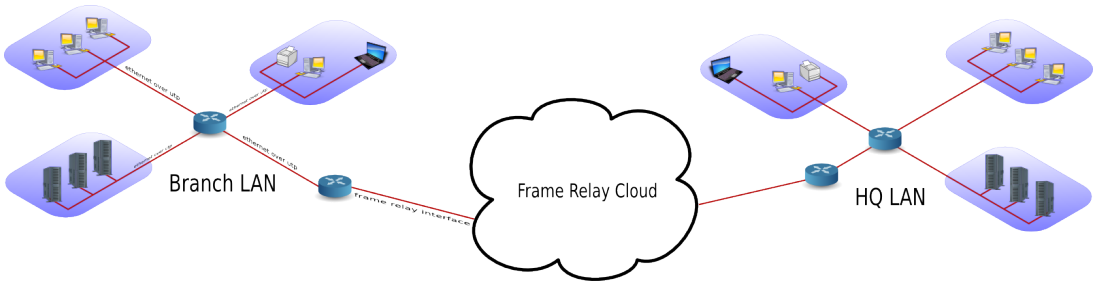
17.3.2. man

A `man` (Metropolitan Area Network) is something inbetween a `lan` and a `wan`, often comprising several buildings on the same campus or in the same city. A `man` can use `fddi` or `ethernet` or other protocols for connectivity.

17.3.3. wan

A `wan` (Wide Area Network) is a network with a lot of distance between the computers (or hosts). These hosts are often connected by `leased lines`. A `wan` does not use `ethernet`, but protocols like `fddi`, `frame relay`, `ATM` or `X.25` to connect computers (and networks).

The picture below shows a branch office that is connected through `Frame Relay` with headquarters.



The acronym `wan` is also used for large surface area networks like the `internet`.

`Cisco` is known for their `wan` technology. They make `routers` that connect many `lan` networks using `wan` protocols.

17.3.4. pan-wpan

Your home network is called a `pan` (Personal Area Network). A wireless `pan` is a `wpan`.

17.4. internet - intranet - extranet

The `internet` is a global network. It connects many networks using the `tcp/ip` protocol stack.

The origin of the `internet` is the `arpanet`. The `arpanet` was created in 1969, that year only four computers were connected in the network. In 1971 the first `e-mail` was sent over the `arpanet`. `E-mail` took 75 percent of all `arpanet` traffic in 1973. 1973 was also the year `ftp` was introduced, and saw the connection of the first European countries (Norway and UK). In 2009 the internet was available to 25 percent of the world population. In 2011 it is estimated that only a quarter of internet webpages are in English.

An `intranet` is a private `tcp/ip` network. An `intranet` uses the same protocols as the `internet`, but is only accessible to people from within one organization.

An `extranet` is similar to an `intranet`, but some trusted organizations (partners/clients/suppliers/...) also get access.

17.5. tcp/ip

17.5.1. history of tcp/ip

In the Sixties development of the `tcp/ip` protocol stack was started by the US Department of Defense. In the Eighties a lot of commercial enterprises developed their own protocol stack: IBM created `sna`, Novell had `ipx/spx`, Microsoft completed `netbeui` and Apple worked with `appletalk`. All the efforts from the Eighties failed to survive the Nineties. By the end of the Nineties, almost all computers in the world were able to speak tcp/ip.

In my humble opinion, the main reason for the survival of `tcp/ip` over all the other protocols is its openness. Everyone is free to develop and use the tcp/ip protocol suite.

17.5.2. rfc (request for comment)

The protocols that are used on the internet are defined in `rfc's`. An rfc or `request for comment` describes the inner working of all internet protocols. The `IETF` (Internet Engineering Task Force) is the sole publisher of these protocols since 1986.

The official website for the rfc's is `http://www.rfc-editor.org`. This website contains all rfc's in plain text, for example rfc2132 (which defines dhcp and bootp) is accessible at http://www.rfc-editor.org/rfc/rfc2132.txt.

17.5.3. many protocols

For reliable connections, you use `tcp`, whereas `udp` is connectionless but faster. The `icmp` error messages are used by `ping`, multicast groups are managed by `igmp`.

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

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

17.5.4. many services

Network cards are uniquely identified by their `mac address`, hosts by their `ip address` and applications by their `port number`.

Common application level protocols like smtp, http, ssh, telnet and ftp have fixed `port numbers`. There is a list of `port numbers` in `/etc/services`.

```
student@linux:~$ grep ssh /etc/services
ssh             22/tcp                  # SSH Remote Login Protocol
ssh             22/udp
```

18. network configuration

(Written by Bert Van Vreckem, https://github.com/bertvv)

This chapter explains how to configure network interfaces on a Linux system. There are considerable differences between distributions, even within the same family. Some distribution even support several methods to configure network interfaces.

We will discuss the most common tools and configuration files of the Debian, and Enterprise Linux families.

If you want to know more about the subject, we recommend to look up the documentation that is specific to your distribution. When you Google, chances are that the information that you find is out of date not relevant for your specific situation.

- Arch Linux:
  - Network Configuration
  - systemd-networkd
- Debian:
  - NetworkConfiguration on the Debian Wiki
  - Network setup in the Debian Reference
- Enterprise Linux: see the Red Hat Enterprise Linux 9 documentation:
  - Configuring and managing networking
  - Configuring an Ethernet connection by using nmcli
- Ubuntu:
  - Configuring Networks

18.1. to gui or not to gui

Recent Linux desktop distributions often provide a GUI to configure the network. Some people complain that these applications mess networking configurations up when used simultaneously with command line configurations. Since the goal of this course is `server` administration, we will assume our Linux servers are always administered through the command line and/or configuration files.

18.2. components of the network configuration

In order to allow a Linux system to communicate with other hosts on a TCP/IP network, it needs the following three settings:

1. An IP address and subnet mask
2. A default gateway
3. One or more DNS servers that can resolve domain names to IP addresses

Usually, those are provided by a DHCP server, but they can also be configured manually.

In the following sections, we will discuss how to check these settings and how to configure them.

18.2.1. checking the network configuration

You can check the network configuration of a Linux system with the `ip` command. The `ip` command is part of the `iproute2` package, which is installed by default on most Linux distributions. The `ip` command is a powerful tool to configure and manage network interfaces, routing, and tunnels. Be sure to read the man-page of `ip(8)` and `ip-address(8)` for detailed information.

If you find a guide or HOWTO online that discusses the `ifconfig` command, it is outdated. The `ifconfig` command is deprecated and should not be used anymore. The `ip` command is the modern replacement for `ifconfig`.

This is the network configuration of a VirtualBox VM with an NAT interface and a host-only interface.:

```
student@linux:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default ql
    link/ether 08:00:27:e6:f5:c9 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
       valid_lft 79579sec preferred_lft 79579sec
    inet6 fe80::a00:27ff:fee6:f5c9/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default ql
    link/ether 08:00:27:0a:46:36 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.56.21/24 brd 192.168.56.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0a:4636/64 scope link
       valid_lft forever preferred_lft forever
```

The `ip address` command (or `ip a` in short) shows the network configuration of the system. The output shows three network interfaces: `lo`, `eth0`, and `eth1`.

- `lo` is the loopback interface, which is used for communication within the system and always has IPv4 address *127.0.0.1/8* and IPv6 address *::1*.
- `eth0` is an Ethernet interface with the IPv4 address *10.0.2.15/24* and IPv6 address *fe80::a00:27ff:fee6:f5c9/64*.
- `eth1` is also an Ethernet interface with the IPv4 address *192.168.56.21/24* and IPv6 address *fe80::a00:27ff:fe0a:4636/64*.

The `UP` flag indicates that the interface is up and running. The `LOWER_UP` flag indicates that the link layer is up and running. If an interface is down, you would see the following:

```
student@linux:~$ ip a show dev eth1
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group defaul
    link/ether 08:00:27:0a:46:36 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.56.21/24 brd 192.168.56.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0a:4636/64 scope link
       valid_lft forever preferred_lft forever
```

The `NO-CARRIER` flag indicates that there is no carrier signal on the interface. This means that there is no network cable connected to the interface, or the cable is broken.

The `ip` command has many subcommands and options. We will discuss some of them in the next sections, but be sure to check the manpage of the `ip(8)` command for more information. Useful options include:

- `-br` or `-brief` show output in a brief format
- `-4` or `-family inet` show only IPv4 addresses
- `-6` or `-family inet6` show only IPv6 addresses
- `-c` or `-color` colorize the output (highlights the interface names and addresses)
- Adding `show dev <interface>` to the command only shows the configuration of the specified interface

18.2.2. network interface names

Traditionally, network interfaces on Linux systems were named `eth0`, `eth1`, `eth2`, etc. Unfortunately, when you install and configure a new baremetal server system with multiple network interfaces, it is not predictable which interface will get which name. If network interfaces are added or removed, the names of the remaining interfaces might change. This can be confusing.

On modern Linux systems, network interface names are provided by the `systemd` predictable network interface names feature. This feature assigns names to network interfaces based on their physical location on the system. The names are based on the firmware or BIOS topology information. This means that the names of the network interfaces are predictable and stable.

It is still possible to disable this feature and revert to the traditional naming scheme, but this is beyond the scope of this course and we do not recommend this.

More information about the network device naming scheme can be found in the manpage `man systemd.net-naming-scheme`.

- a name starting with `en` denotes an Ethernet device
- a name starting with `wl` denotes a wireless LAN device
- the rest of the name is based on the physical location of the device, e.g.
  - `enp0s3` stands for “Ethernet device, PCI bus 0, slot 3”
  - `wlp3s0` stands for “Wireless LAN device, PCI bus 3, slot 0”

Remark that in the example above, the `altname` shows the predictable name of the network interfaces. The actual name of network interface `eth0` is `enp0s3` and `eth1` is in fact `enp0s8`.

18.2.3. routing table and default gateway

The routing table of a Linux system can be shown with the `ip route` or `ip r` command. The routing table shows the available routes and the default gateway.

```
student@linux:~$ ip route
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
192.168.56.0/24 dev eth1 proto kernel scope link src 192.168.56.21
```

This is a typical routing table for a server system. The first line shows the default route, i.e. the IP address of the router that connects the LAN to the Internet. In the example above, the default gateway is *10.0.2.2* and the interface to reach the default gateway is *eth0*.

The other two lines correspond with the two network interfaces of the system. Packets that are sent to destinations within the subnet of the network interfaces can be delivered directly.

18.2.4. DNS configuration

The final part of the network configuration of a Linux system is the DNS configuration, i.e. a list of DNS servers that can be used to resolve domain names to IP addresses. The way DNS servers are configured depends on your Linux distribution and/or its version.

Traditionally, DNS servers were kept in the file `/etc/resolv.conf`:

```
student@linux:~$ cat /etc/resolv.conf
domain home
search home
nameserver 10.0.2.3
```

In this example, the DNS server has IP address 10.0.2.3 (which is typical for a VirtualBox NAT interface). The `domain` and `search` lines are used to append the domain name to hostnames that are not fully qualified.

However, on recent versions of several Linux distribututions, the `/etc/resolv.conf` file is often managed by the `systemd-resolved` service. This service is part of the `systemd` suite and provides network name resolution to local applications. The `systemd-resolved` service is configured through the file `/etc/systemd/resolved.conf` and the `/etc/resolv.conf` file is a symlink to `/run/systemd/resolve/stub-resolv.conf`.

The following example is a typical `/etc/resolv.conf` file on a system with `systemd-resolved`:

```
student@linux:~$ cat /etc/resolv.conf
# This is /run/systemd/resolve/stub-resolv.conf managed by man:systemd-
resolved(8).
# Do not edit.
nameserver 127.0.0.53
options edns0 trust-ad
search home
student@linux:~$ ls -l /etc/resolv.conf
lrwxrwxrwx 1 root root 39 Jan 21 19:42 /etc/resolv.conf -> ../run/systemd/resolve/stub-
resolv.conf
```

The DNS server is set to 127.0.0.53, which seems to be a combination of the loopback address 127.0.0.1 and the port number 53 of the DNS service. Any IP address in the 127.0.0.0/8 range is considered to be a loopback address and will behave just like 127.0.0.1. `systemd-resolved` listens on a server socket with port 53 connected to the loopback interface.

In the example below, we use the Show Sockets (`ss`) command to verify that `systemd-resolved` is listening on UDP port 53 and next, we send a DNS query to the IP address specified in `/etc/resolv.conf` (see below for more info on the `dig` command that is used here):

```
student@linux:~$ sudo ss -ulnp  | grep 'resolve'
UNCONN 0      0      127.0.0.53%lo:53         0.0.0.0:*    users:(("systemd-
resolve",pid=582,fd=13))
student@linux:~$ dig +short @127.0.0.53 www.linux-training.be
188.40.26.208
```

This means that your Linux system is providing its own DNS service. However, it can't make up the IP addresses of hosts on the Internet by itself, it needs to forward the request to a real DNS server. You can check which DNS servers are used by `systemd-resolved` with the `resolvectl` command:

```
student@linux:~$ resolvectl dns
$ resolvectl dns
Global:
Link 2 (enp0s3): 195.130.131.1 195.130.130.1
Link 3 (enp0s8):
```

In this example, the first network interface uses the DNS servers 195.130.131.1 and 195.130.130.1 (which are operated by the Belgian ISP Telenet). The second network interface is not connected to the Internet and does not have any DNS servers.

18.3. verifying network connectivity

18.3.1. ping

In order to check whether a Linux system can communicate with other hosts on the network, you can use the `ping` command. The `ping` command sends ICMP echo request packets to a host and waits for ICMP echo reply packets. If the host is reachable, it will respond to the echo request packets.

On a Linux system, the `ping` command will continue to send echo request packets until you stop it with `Ctrl-C`. You can also limit the number of echo request packets with the `-c` option.

```
student@linux:~$ ping -c 3 10.0.2.3
PING 10.0.2.3 (10.0.2.3) 56(84) bytes of data.
64 bytes from 10.0.2.3: icmp_seq=1 ttl=64 time=1.18 ms
64 bytes from 10.0.2.3: icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from 10.0.2.3: icmp_seq=3 ttl=64 time=0.835 ms

--- 10.0.2.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.835/1.063/1.181/0.161 ms
```

You can specify either an IP address or a hostname:

```
student@linux:~$ ping -c 3 www.linux-training.be
PING www.linux-training.be (188.40.26.208) 56(84) bytes of data.
64 bytes from www115.your-server.de (188.40.26.208): icmp_seq=1 ttl=50 time=28.8 ms
64 bytes from www115.your-server.de (188.40.26.208): icmp_seq=2 ttl=50 time=29.2 ms
64 bytes from www115.your-server.de (188.40.26.208): icmp_seq=3 ttl=50 time=28.1 ms

--- www.linux-training.be ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 28.063/28.699/29.187/0.470 ms
```

If IPv6 is enabled on your system, you can also use the `-6` option or the `ping6` command to send echo request packets to an IPv6 address.

```
student@linux:~$ ping6 -c 3 www.linux-training.be
PING www.linux-training.be(www115.your-server.de (2a01:4f8:d0a:1044::2)) 56 data bytes
64 bytes from www115.your-server.de (2a01:4f8:d0a:1044::2): icmp_seq=1 ttl=51 time=23.9 ms
64 bytes from www115.your-server.de (2a01:4f8:d0a:1044::2): icmp_seq=2 ttl=51 time=23.9 ms
64 bytes from www115.your-server.de (2a01:4f8:d0a:1044::2): icmp_seq=3 ttl=51 time=21.6 ms

--- www.linux-training.be ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 21.595/23.138/23.911/1.091 ms
```

18.3.2. traceroute

The `ping` command can tell you whether routing to a host is possible. The `traceroute` command is goes a step further and also shows you which route the packets take to reach their destination. It does this by sending packets with an increasing time-to-live (TTL) value. The IP standard states that a router must decrease the TTL value of a packet by 1. If the TTL value reaches 0, the packet is discarded and the router sends an *ICMP time exceeded* message back to the sender. The `traceroute` command uses this mechanism to find out which routers are on the path to the destination.

Originally, `traceroute` sent an UDP packet to some unlikely port on the target system. If the target system was reached, it would send an ICMP port unreachable message back. Unfortunately, nowadays many routers are configured to silently drop packets to unlikely ports, so this method is not reliable anymore.

In the following example, we use the `traceroute` command to find out which routers are on the path to the web server of the Linux Training website. Unfortunately, we don't get any useful information:

```
student@linux:~$ traceroute www.linux-training.be
traceroute to www.linux-training.be (188.40.26.208), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.443 ms  1.023 ms  0.866 ms
 2  * * *
 3  * * *
... (many lines omitted) ...
29  * * *
30  * * *
```

Fortunately, the `traceroute` command provides some options to select a different method, e.g. the `-I` option to use *ICMP echo request* packets (as used by `ping`) and the `-T` option to use TCP SYN packets to port 80 (http, as if you want to request a web page from the target host). You will need to try different methods to find out which one works best in your situation. Remark that you need root privileges to use the `-I` and `-T` options.

```
student@linux:~$ traceroute -I www.linux-training.be
You do not have enough privileges to use this traceroute method.
socket: Operation not permitted
student@linux:~$ sudo traceroute -I www.linux-training.be
traceroute to www.linux-training.be (188.40.26.208), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  1.251 ms  1.001 ms *
 2  192.168.0.1 (192.168.0.1)  6.929 ms * *
 3  94-224-72-1.access.telenet.be (94.224.72.1)  17.122 ms * *
 4  dD5E0CB8A.access.telenet.be (213.224.203.138)  22.673 ms * *
 5  * * *
 6  4.68.70.53 (4.68.70.53)  49.133 ms  16.993 ms  16.451 ms
 7  * * *
 8  62.67.26.138 (62.67.26.138)  25.287 ms  26.677 ms  26.513 ms
 9  core11.nbg1.hetzner.com (213.239.245.73)  26.374 ms  30.749 ms  30.598 ms
10  ex9k2.dc1.nbg1.hetzner.com (213.239.203.214)  30.455 ms  35.780 ms  35.636 ms
11  www115.your-server.de (188.40.26.208)  36.502 ms  37.340 ms  34.059 ms
```

When `traceroute` receives a packet from a router, it will try to do a reverse DNS lookup to find out its host name. These are useful (as you can sometimes deduce info about the router's location), but may take some time to resolve. In that case, you can use the `-n` option and immediately show only the IP addresses.

18.3.3. tracepath

The `tracepath` command is a simplified version of `traceroute`. It uses the same strategy as `traceroute` to find routers along the path to the destination. It does not require root privileges, but it also does not provide the alternative methods that `traceroute` does. That means that `tracepath` is less likely to work than `traceroute`, but it is worth a try if you don't have root privileges.

```
$ tracepath www.linux-training.be
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              3.430ms
 1:  _gateway                                              5.459ms
 2:  no reply
...(many lines omitted)...
29:  no reply
30:  no reply
     Too many hops: pmtu 1500
```

18.3.4. name resolution

You can test whether name resolution is available for your system with e.g. the `dig` command:

```
student@linux:~$ dig +short @10.0.2.3 www.linux-training.be
188.40.26.208
```

The argument `@10.0.2.3` tells `dig` to send the request to that specific DNS server. If you omit it, the system will use the DNS server configured in `/etc/resolv.conf`.

Windows users will probably be more familiar with the `nslookup` command, but it is much more limited than `dig`.

```
vagrant@debian:~$ nslookup www.linux-training.be 10.0.2.3
Server:         10.0.2.3
Address:        10.0.2.3#53

Non-authoritative answer:
Name:   www.linux-training.be
Address: 188.40.26.208
Name:   www.linux-training.be
Address: 2a01:4f8:d0a:1044::2
```

If the command `dig` and `nslookup` are not available, and you can't install them (because of network issues, for example!), you can fall back to the `getent ahosts` command to check whether DNS resolution works:

```
student@linux:~$ getent ahosts www.linux-training.be
188.40.26.208   STREAM www.linux-training.be
188.40.26.208   DGRAM
188.40.26.208   RAW
2a01:4f8:d0a:1044::2 STREAM
2a01:4f8:d0a:1044::2 DGRAM
2a01:4f8:d0a:1044::2 RAW
```

18.3.5. arp (ip neighbor)

On a local network, IP addresses must be translated to MAC addresses in order to send packets to other hosts. This translation is done with the layer two broadcast protocol ARP (Address Resolution Protocol).

Your system will keep a cache of the MAC addresses of the hosts it has recently communicated with. You can view this cache with the `ip neighbor` (or `neighbour`, or `n`) command:

```
[student@linux ~]$ ip neighbour
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
192.168.56.12 dev eth1 lladdr 08:00:27:0a:46:36 STALE
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 DELAY
```

The `ip neighbour` command shows the IP address, the network interface, the MAC address, and the state of the entry. The state can be one of the following:

- `REACHABLE`: the MAC address is known and the system has recently communicated with the host
- `STALE`: the MAC address is known, but the system has not recently communicated with the host
- `DELAY`: the MAC address is not known and the system is trying to find it. If you see this state, it means that the system is trying to send an ARP request to find the MAC address of the host. If the host exists, the state will change to `REACHABLE` after a while.

The `ip n` command also allows you to edit the ARP cache, e.g. to add a static entry or to remove an entry.

To get a specific entry:

```
[student@linux ~]$ ip n get 10.0.2.3 dev eth0
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
```

To delete a specific entry (requires root privileges), use `ip n del`:

```
[student@el ~]$ ip n del 10.0.2.3 dev eth0
RTNETLINK answers: Operation not permitted
[student@el ~]$ sudo ip n del 10.0.2.3 dev eth0
[student@el ~]$ ip n
192.168.56.12 dev eth1 lladdr 08:00:27:0a:46:36 STALE
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
```

To delete all entries for a specific interface (requires root privileges), use `ip n flush`:

```
[student@el ~]$ ip n
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
192.168.56.12 dev eth1 lladdr 08:00:27:0a:46:36 STALE
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
[student@el ~]$ ip n flush dev eth1
Failed to send flush request: Operation not permitted
[student@el ~]$ sudo !!
sudo ip n flush dev eth1
[student@el ~]$ ip n
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
```

See the man-page of `ip-neighbour(8)` for more information.

18.3.6. what is my public IP address?

Quite often, you find yourself behind a NAT router, and you have an IP address within a privat range (e.g. 192.168.0.0/24, 172.16.0.0/16 or 10.0.0.0/8). If you want to know what your public IP address is, you can use the `curl` command to get this information from a website that provides this service. The following example uses the `ifconfig.me` website (you can also use `icanhazip.com`):

```
student@linux:~$ curl ifconfig.me
94.224.76.64
student@linux:~$ dig +short -x 94.224.76.64
94-224-76-64.access.telenet.be.
```

You can try the same with the `ipinfo.io` website, which gives some more information in JSON format:

```
student@linux:~$ curl ipinfo.io
{
  "ip": "94.224.76.64",
  "hostname": "94-224-76-64.access.telenet.be",
  "city": "Dilbeek",
  "region": "Flanders",
  "country": "BE",
  "loc": "50.8480,4.2597",
  "org": "AS6848 Telenet BV",
  "postal": "1700",
  "timezone": "Europe/Brussels",
  "readme": "https://ipinfo.io/missingauth"
}
```

18.4. configuring network settings

The configuration files that keep the network settings differ between Linux distributions. In this section, we will discuss the configuration of Debian and Enterprise Linux (Red Hat-like) systems.

18.4.1. temporary changes with the ip command

On both Debian and Enterprise Linux, and many other Linux distribution, you can use the `ip` command to configure network interfaces. However, the changes you make with the `ip` command are not persistent. If you reboot the system, the changes will be lost. To make the changes persistent, you need to edit the configuration files, as we will discuss in the next sections.

To change the IP address of an interface, you can use the `ip address add` or `replace` command. For example, to change the IP address of `eth1` to 192.168.56.99:

```
[vagrant@el ~]$ ip a replace 192.168.56.99 dev eth1
[vagrant@el ~]$ ip -4 -br a show dev eth1
eth1       UP             192.168.56.9/24 192.168.56.99/32
```

There's also a `del` option to remove an IP address from an interface:

```
[vagrant@el ~]$ ip a del 192.168.56.99 dev eth1
[vagrant@el ~]$ ip -4 -br a show dev eth1
eth1       UP             192.168.56.9/24
```

18.4.2. /etc/network/interfaces (Debian)

On Debian-based systems (including Ubuntu, Linux Mint, Raspberry Pi OS, etc.), the network settings are kept in the `/etc/network/interfaces` file. This file is used by the `ifup` and `ifdown` commands to configure and deconfigure network interfaces. The `ifup` and `ifdown` commands are part of the `ifupdown` package, which is installed by default on Debian-based systems. For more information about this configuration file, see its manpage with `man 5 interfaces`.

An example of a network configuration file for a system with two network interfaces, one with a dynamic IP address and one with a static IP address, is shown below:

```
student@debian:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp
pre-up sleep 2

auto eth1
iface eth1 inet static
      address 192.168.56.21
      netmask 255.255.255.0
```

In this example, we find configuration for three network interfaces: `lo`, `eth0` and `eth1`.

- The `lo` interface is the loopback interface, which is always configured with the `loopback` keyword.
  - The `auto` keyword means that the interface will be brought up automatically at boot time.
- The `eth0` interface is configured to use DHCP to get an IP address.
  - The `allow-hotplug` keyword means that the interface will be brought up automatically when various condition changes are detected.
  - The `pre-up` keyword means that the command that follows will be executed before the interface is brought up. In this case, the `sleep 2` command is used to wait for 2 seconds before the interface is brought up. This is useful when the interface is connected to a switch that needs some time to come up.
- The `eth1` interface is configured with a static IP address (192.168.56.21) and network mask (255.255.255.0 or /24).

After you have made changes to the `/etc/network/interfaces` file, you can apply the changes with the `ifup` and `ifdown` commands.

For example, let's change the IP address on `eth1` to 192.168.56.12

```
student@debian:~$ sudo nano /etc/network/interfaces
... (make the necessary changes to the file) ...
student@debian:~$ sudo ifdown eth1
RTNETLINK answers: Cannot assign requested address
```

```
student@debian:~$ sudo ifup eth1
student@debian:~$ ip -br a show dev eth1
eth1      UP      192.168.56.12/24 fe80::a00:27ff:fe0a:4636/64
```

18.4.3. /etc/sysconfig/network-scripts (Enterprise Linux)

On Enterprise Linux (Red Hat, Fedora, AlmaLinux, CentOS, etc.), the network settings are traditionally kept in the `/etc/sysconfig/network-scripts` directory. Each network interface has its own configuration file in this directory. The configuration files are named `ifcfg-<interface>`, where `<interface>` is the name of the network interface.

Remark that Red Hat has been moving away from this system since RHEL 7 and is migrating to the *NetworkManager* service. However, for now, the old configuration files are still present and can be used. See the next section for more information on editing network settings with *NetworkManager*.

An example of a network configuration file for a EL9 system with two network interfaces, one with a dynamic IP address and one with a static IP address, is shown below:

```
[student@el ~]$ ls /etc/sysconfig/network-scripts/
ifcfg-eth0  ifcfg-eth1  readme-ifcfg-rh.txt
[student@el ~]$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0
DEVICE=eth0
ONBOOT=yes
[student@el ~]$ cat /etc/sysconfig/network-scripts/ifcfg-eth1
NM_CONTROLLED=yes
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.56.11
NETMASK=255.255.255.0
DEVICE=eth1
PEERDNS=no
```

The important settings in these files are:

- `NAME` and `DEVICE`: the name of the network interface
- `ONBOOT=yes`: the interface will be brought up automatically at boot time
- `BOOTPROTO`: the method to get an IP address:
  - `dhcp`: the interface will get an IP address from a DHCP server
  - `none`: the interface will use a static IP address
- `IPADDR` and `NETMASK`: the static IP address and network mask

After you have made changes to the configuration files in `/etc/sysconfig/network-scripts`, you can apply the changes with the `nmcli` (NetworkManager Command-Line Interface) command.

For example, let's change the IP address on `eth1` to 192.168.56.9

```
[student@el ~]$ ip -br a show dev eth1
eth1             UP             192.168.56.11/24 fe80::a00:27ff:fec8:fbc4/64
[student@el ~]$ sudo vi /etc/sysconfig/network-scripts/ifcfg-eth1
... (make the necessary changes to the file) ...
[student@el ~]$ sudo nmcli connection reload
[student@el ~]$ nmcli connection show
NAME         UUID                                  TYPE      DEVICE
eth0         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  eth1
lo           7aec5c7a-0122-4c27-a48b-0755c9cf0769  loopback  lo
[student@el ~]$ sudo nmcli connection up 'System eth1'
[student@el ~]$ ip -br a show dev eth1
eth1      UP       192.168.56.9/24 fe80::a00:27ff:fec8:fbc4/64
```

18.4.4. NetworkManager (EL>=7, Fedora, Mint, ...)

On recent versions of RHEL (since RHEL 7) and Fedora, and even Debian and derivatives, *NetworkManager* is used to manage network connections. The *NetworkManager* service is a dynamic network control and configuration system that attempts to keep network devices and connections up and active when they are available. It manages Ethernet, WiFi, mobile broadband (WWAN), and PPPoE devices, and provides VPN integration with a variety of different VPN services.

An argument can be made that this is quite useful for laptops, who often move between networks and need to switch between wired and wireless connections. However, for servers, it is often considered overkill and unnecessary. Nevertheless, *NetworkManager* is installed and actiated on many systems, so it is necessary to know how to use it.

In this section, we'll discuss `nmcli`, the command-line interface to *NetworkManager*. There are alternative ways to configure *NetworkManager* (e.g. with the `nmtui` command or the graphical user interface), but we will focus on `nmcli` here. Reason is that `nmcli`-based configuration can be automated in scripts, while actions in an interactive user interface cannot.

With *NetworkManager*, you can provide a connection profile for each network interface. A connection profile is a set of properties that describe how the network interface should be configured. The connection profiles are stored in the `/etc/NetworkManager/system-connections` directory.

To show the connection profiles, use the `nmcli connection show` command. If you want to see specific details of a connection profile, use the `nmcli connection show <name>` command.

```
[student@el ~]$ nmcli connection show
NAME         UUID                                  TYPE      DEVICE
eth0         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
lo           7aec5c7a-0122-4c27-a48b-0755c9cf0769  loopback  lo
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  eth1
[student@el ~]$ nmcli connection show eth0
connection.id:                          eth0
connection.uuid:                        5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              eth0
connection.autoconnect:                 yes
... (more details of the connection profile for eth0) ...
[student@el ~]$ nmcli device show eth1
... (more details of the network device eth1) ...
```

There's a lot of output, so you may want to use the `--fields` or `-f` option to limit the output to the fields you are interested in.

```
[student@el ~]$ nmcli -f IP4 device show eth0
IP4.ADDRESS[1]:         10.0.2.15/24
IP4.GATEWAY:            10.0.2.2
IP4.ROUTE[1]:           dst = 0.0.0.0/0, nh = 10.0.2.2, mt = 102
IP4.ROUTE[2]:           dst = 10.0.2.0/24, nh = 0.0.0.0, mt = 102
IP4.DNS[1]:             10.0.2.3
IP4.DOMAIN[1]:          home
[student@el ~]$ nmcli -f IP6 dev sh eth1
IP6.ADDRESS[1]:         fe80::a00:27ff:fec8:fbc4/64
IP6.GATEWAY:            --
IP6.ROUTE[1]:           dst = fe80::/64, nh = ::, mt = 256
```

In this example, the names of the connection profiles are `eth0`, `lo` and `System eth1`. The name of the last one is not consistent with the other two, but that can be changed!

```
[student@el ~]$ sudo nmcli con modify 'System eth1' connection.id eth1
[student@el ~]$ nmcli con sh
NAME    UUID                                  TYPE      DEVICE
eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
eth1    9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  eth1
lo      7aec5c7a-0122-4c27-a48b-0755c9cf0769  loopback  lo
```

To configure IPv4 settings for a connection profile, use the `nmcli connection modify` command. The following example shows how to configure the `eth1` connection profile with a static IP address:

```
[student@el ~]$ sudo nmcli connection modify eth1 \
  ipv4.method manual ipv4.addresses 192.168.56.9/24
```

You can also configure the default gateway and DNS servers (if appropriate for that connection):

```
[student@el ~]$ sudo nmcli connection modify eth1 \
  ipv4.method manual \
  ipv4.addresses 192.168.56.9/24 ipv4.gateway 192.168.56.254 \
  ipv4.dns 192.168.56.254 ipv4.dns-search example.com
```

To use DHCP instead, enter:

```
[student@el ~]$ sudo nmcli connection modify eth1 ipv4.method auto
```

To configure IPv6 settings and use stateless address autoconfiguration (SLAAC), use the following command:

```
[student@el ~]$ sudo nmcli connection modify eth1 ipv6.method auto
```

This is the default, so it may not be necessary to set it explicitly.

To set a static IPv6 address, network mask, default gateway, DNS servers and DNS search domain, use the following command (modifying the IPv6 addresses to ones that are appropriate for your network):

```
[student@el ~]$ sudo nmcli connection modify eth1 \
  ipv6.method manual \
  ipv6.addresses 2001:db8:1::9/64 ipv6.gateway 2001:db8:1::1 \
  ipv6.dns 2001:db8:1::1 ipv6.dns-search example.com
```

Finally, to activate the profile, enter:

```
[student@el ~]$ sudo nmcli connection up eth1
```

Verify the changes with `ip a`, `ip r` and `resolvectl dns` commands (or the `/etc/resolv.conf` file).

18.4.5. Netplan (Ubuntu)

Recent versions of Ubuntu use *Netplan* to configure network interfaces. *Netplan* is a utility that converts a declarative description of the desired network configuration in YAML format to the required format for the underlying network management system, either `systemd-networkd` (the default for Ubuntu) or `NetworkManager`.

The configuration files are stored in the `/etc/netplan` directory. This directory may contain one or more YAML files starting with a double digit number, e.g. `01-netcfg.yaml` (or similar).

```
student@ubuntu:~$ cat /etc/netplan/01-netcfg.yaml
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: true
```

In this example, we have a single interface that is configured to get an IP address from a DHCP server. If you want to use a static IP address, you can use the following configuration:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses:
        - 10.0.2.15/24
      routes:
        - to: default
          via: 10.0.2.2
      nameservers:
        search:
          - home
        addresses:
          - 10.0.2.3
```

In this example we reconstructed the settings a VirtualBox VM would get if it attaches to a NAT adapter.

After you have made changes to the configuration file in `/etc/netplan`, you can apply the changes with the `sudo netplan apply` command.

18.4.6. changing the hostname

The hostname of a Linux system is a name given to it by the system administrator. In many cases, this is just a name for internal use in the logs and prompts. If you want other hosts on the network to be able to reach your system by its hostname, you will need register it in the DNS server for the domain.

On modern Linux distributions, managing the host name is done by `systemd-hostnamed`, part of the `systemd` suite. The `hostnamectl` command is used to query and change the system hostname and related settings. Read the man-page of `hostnamectl(1)` for detailed information.

Without any arguments, `hostnamectl` will show the current settings, e.g.:

```
[student@el ~]$ hostnamectl
 Static hostname: el
       Icon name: computer-vm
         Chassis: vm
      Machine ID: 8616fe49eae5443aac94b55664267068
         Boot ID: 47762feaf5614f33a6bd7576374e186e
  Virtualization: oracle
Operating System: AlmaLinux 9.3 (Shamrock Pampas Cat)
     CPE OS Name: cpe:/o:almalinux:almalinux:9::baseos
          Kernel: Linux 5.14.0-362.13.1.el9_3.x86_64
    Architecture: x86-64
 Hardware Vendor: innotek GmbH
  Hardware Model: VirtualBox
Firmware Version: VirtualBox
```

This example is from Ubuntu 22.04 in Windows Subsystem For Linux:

```
$ hostnamectl
 Static hostname: NB23-G89HHX3
       Icon name: computer-container
         Chassis: container
      Machine ID: 0444ccbadb2245f4a0caac3a81a7d7cf
         Boot ID: fa53f951d31f41a3ab3935429b9b59a2
  Virtualization: wsl
Operating System: Ubuntu 22.04.3 LTS
          Kernel: Linux 5.15.146.1-microsoft-standard-WSL2
    Architecture: x86-64
```

To change the hostname, use the `hostnamectl set-hostname` command:

```
[student@el ~]$ sudo hostnamectl set-hostname alma9
[student@el ~]$
```

Remark that the prompt hasn't changed yet! You will need to log out and log back in to see the new hostname in the prompt.

`systemd-hostnamed` distinguishes three different hostnames: the high-level "pretty" hostname which might include all kinds of special characters (e.g. "Lennart's Laptop"), the "static" hostname which is the user-configured hostname (e.g. "lennarts-laptop"), and the transient hostname which is a fallback value received from network configuration (e.g. "node12345678"). If a static hostname is set to a valid value, then the transient hostname is not used.

18.4.7. changing the MAC address

The MAC address of a network interface is a unique identifier assigned to a network interface for communications at the data link layer of a network segment. It is usually assigned by the manufacturer of the network interface. However, it is possible to change the MAC address of a network interface with the `ip` command.

To change the MAC address of a network interface, use the `ip link set` command. You will need to have root privileges to do this.

```
[student@el ~]$ ip link show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT gro
    link/ether 08:00:27:c8:fb:c4 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
[student@el ~]$ sudo ip link set eth1 address 08:00:de:ad:be:ef type ether
[student@el ~]$ ip link show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT gro
   link/ether 08:00:de:ad:be:ef brd ff:ff:ff:ff:ff:ff permaddr 08:00:27:c8:fb:c4
    altname enp0s8
```

Remark that the original MAC address is still visible with the `permaddr` keyword.

See the man-page of `ip-link(8)` for more information.

18.5. practice: network configuration

The best way to learn this material is by setting up a multi-VM lab environment and practicing the commands. We suggest to use Vagrant to create several VMs that are all connected to a host-only or internal network. With the instructions below, you will create an Ubuntu and a Fedora VM, each with an internal network interface. These won't be configured out-of-the-box, so the goal of exercises below is to allow both VMs to communicate with each other by configuring the network interfaces.

18.5.1. lab setup

To set up the lab environment, follow these steps:

1. Ensure Vagrant and VirtualBox (and, optionally, Git) are installed on your system.
2. Download the code of Github repository bertvv/vagrant-shell-skeleton (as a .zip or by cloning the repository). Give the directory a name that reflects the purpose of the lab environment, e.g. `network-config-lab`.
3. Edit the file `vagrant-hosts.yml` and add the following content:

```
- name: ubuntu
  box: bento/ubuntu-22.04 # Or a more recent LTS version if available
  intnet: true
  auto_config: false
- name: fedora
  box: bento/fedora-latest
  intnet: true
  auto_config: false
```

   The existing entry can be removed or commented out. Check the changes with `vagrant status`.

```
> vagrant status
Current machine states:

fedora               not created (virtualbox)
ubuntu               not created (virtualbox)
```

4. Start the VMs with `vagrant up`. This will create two VMs, `ubuntu` and `fedora`, each with an unconfigured `intnet` network interface. You can log in to the VMs with `vagrant ssh ubuntu` and `vagrant ssh fedora`.

18.5.2. exercises

1. Log in to each VM and look up the following information:
   - MAC addresses of each network interface
   - IPv4 and IPv6 addresses and network masks of each network interface
   - The public IP address of your VMs, lab environment or home network
   - The routing table and default gateway
   - The DNS server(s)
   - ARP address cache
2. Use the results with the information about the lab setup to draw a network diagram and fill out an IP address table that states for each host and network interface the MAC address, IPv4/6 addresses, network masks, and default gateway.

   For now, leave open the values for interfaces that aren't configured yet, but complete the table as you progress through the exercises.

   Never underestimate the value of a good network diagram and an overview table to understand and troubleshoot network issues!
3. Assign a static IP address to the `eth1` interface of each VM. Use the following addresses:
   - Ubuntu: 192.168.100.1/24, fd00:1337:cafe:1::1/64
   - Fedora: 192.168.100.2/24, fd00:1337:cafe:1::2/64
4. Check if the VMs can ping each other. If not, troubleshoot the issue and fix it.
5. Check the ARP cache on each VM. Is there a difference with the previous time you did this?

18.6. solutions: network configuration

18.6.1. network diagram

```
graph LR
    I(Internet):::cloud
    I --- N1[NAT]:::switch
    I --- N2[NAT]:::switch

    N1 --- |eth0| U([ubuntu]):::host
    N2 --- |eth0| F([fedora]):::host

    subgraph "Internal network"
        N3[intnet]:::switch
        U --- |eth1| N3
```

```
        F --- |eth1| N3
    end

    classDef host fill:#A5CA72,stroke:#A5CA72,stroke-width:2px
    classDef switch fill:#4CA2D5,stroke:#000,stroke-width:1px
          classDef   cloud   fill:#eee,strqoke:#000,stroke-width:2px,stroke-
dasharray:10,5
```

- Hosts are the nodes with a green background (the Fedora and Ubuntu VMs).
- Switches/LANs are the rectangular nodes with a dark blue background. In this case, the switches are not real devices, but virtualized networks within VirtualBox.
  - The Ubuntu and Fedora VMs are each connected to the Internet via a separate NAT adapter with interface `eth0` (the predictable name would be `enp0s3`)
  - They are both connected with `eth1` (predictable name `enp0s8`) to a shared internal network.

18.6.2. IP address table

Remark that the MAC and IPv6 addresses will be different on your system. The IPv4 addresses should be the same, as we will configure them, or their value is predictable.

Host	Iface	MAC address	IPv4 address/mask	IPv6 address/mask	GW
ubuntu	eth0	08:00:27:87:8e:29	10.0.2.15/24	fe80::a00:27ff:fe87:8e29/64	10.0.2.2
	eth1	08:00:27:df:a2:e4	192.168.100.1/24	fd00:1337:cafe:1::1/64	–
fedora	eth0	08:00:27:d5:0f:5d	10.0.2.15/24	fe80::e0e:40bf:aa93:ed7d/64	10.0.2.2
	eth1	08:00:27:7e:c1:07	192.168.100.2/24	fd00:1337:cafe:1::2/64	–

18.6.3. solutions

1. Log in to each VM and look up the following information:
   - MAC addresses of each network interface: general command: `ip a`, or specific: `ip -br l show dev eth0` or `eth1`

     On Ubuntu:

     ```
     vagrant@ubuntu:~$ ip a
     1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default ql
         link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
         inet 127.0.0.1/8 scope host lo
         valid_lft forever preferred_lft forever
         inet6 ::1/128 scope host
         valid_lft forever preferred_lft forever
     2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group de
         link/ether 08:00:27:87:8e:29 brd ff:ff:ff:ff:ff:ff
         altname enp0s3
        inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic eth0
         valid_lft 80602sec preferred_lft 80602sec
         inet6 fe80::a00:27ff:fe87:8e29/64 scope link
         valid_lft forever preferred_lft forever
     3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 100
         link/ether 08:00:27:df:a2:e4 brd ff:ff:ff:ff:ff:ff
         altname enp0s8
     vagrant@ubuntu:~$ ip -br l show dev eth0
     eth0             UP             08:00:27:87:8e:29 <BROADCAST,MULTICAST,UP,LOWER_UP>
     ```

```
vagrant@ubuntu:~$ ip -br l show dev eth1
eth1             DOWN           08:00:27:df:a2:e4 <BROADCAST,MULTICAST>
```

On Fedora:

```
[vagrant@fedora ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default ql
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group de
    link/ether 08:00:27:d5:0f:5d brd ff:ff:ff:ff:ff:ff
    altname enp0s3
   inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
    valid_lft 80257sec preferred_lft 80257sec
    inet6 fe80::e0e:40bf:aa93:ed7d/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group de
    link/ether 08:00:27:7e:c1:07 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
[vagrant@fedora ~]$ ip -br l show dev eth0
eth0         UP         08:00:27:d5:0f:5d <BROADCAST,MULTICAST,UP,LOWER_UP>
[vagrant@fedora ~]$ ip -br l show dev eth1
eth1         UP         08:00:27:7e:c1:07 <BROADCAST,MULTICAST,UP,LOWER_UP>
```

- IPv4 and IPv6 addresses and network masks of each network interface

  See the output of `ip a` above. To get specific information, use `ip -br a show dev eth0` or `eth1`.

```
vagrant@ubuntu:~$ ip -br a show dev eth0
eth0         UP         10.0.2.15/24 metric 100 fe80::a00:27ff:fe87:8e29/64
vagrant@ubuntu:~$ ip -br a show dev eth1
eth1             DOWN
```

```
[vagrant@fedora ~]$ ip -br a show dev eth0
eth0           UP           10.0.2.15/24 fe80::e0e:40bf:aa93:ed7d/64
[vagrant@fedora ~]$ ip -br a show dev eth1
eth1             UP             fe80::e3fd:c3c5:a67d:e676/64
```

- The public IP address of your VMs, lab environment or home network

```
[vagrant@fedora ~]$ curl ifconfig.me
94.12.34.56
```

  The result is the same on the Ubuntu VM.

- The routing table and default gateway

```
vagrant@ubuntu:~$ ip r
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
10.0.2.2 dev eth0 proto dhcp scope link src 10.0.2.15 metric 100
10.0.2.3 dev eth0 proto dhcp scope link src 10.0.2.15 metric 100
```

```
[vagrant@fedora ~]$ ip r
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
```

  Remark that there will be no default gateway for `eth1` as it is not connected to the Internet.

- The DNS server(s)

```
vagrant@ubuntu:~$ resolvectl dns
Global:
Link 2 (eth0): 10.0.2.3
Link 3 (eth1):
```

```
[vagrant@fedora ~]$ resolvectl dns
Global:
Link 2 (eth0): 10.0.2.3
Link 3 (eth1):
```

For both VMs, the DNS server is provided by VirtualBox's NAT adapter.

- ARP address cache

```
vagrant@ubuntu:~$ ip n
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 DELAY
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
```

```
[vagrant@fedora ~]$ ip n
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 DELAY
```

2. See completed address table above.
3. Assign a static IP address to the `eth1` interface of each VM. Use the following addresses:

   - Ubuntu: 192.168.100.1/24, fd00:1337:cafe:1::1/64
   - Fedora: 192.168.100.2/24, fd00:1337:cafe:1::2/64

   On Ubuntu:

```
vagrant@ubuntu:~$ ls -l /etc/netplan/
total 8
-rw------- 1 root root 144 Jan 30 06:24 00-installer-config.yaml
-rw------- 1 root root 150 Mar 10 16:02 01-netcfg.yaml`
vagrant@ubuntu:~$ sudo cat /etc/netplan/01-netcfg.yaml
```

   Content of netplan file:

```
network:
version: 2
ethernets:
  eth0:
    dhcp4: true
  # add these lines:
  eth1:
    addresses:
      - 192.168.100.1/24
      - fd00:1337:cafe:1::1/64
```

   Apply the changes:

```
vagrant@ubuntu:~$ sudo netplan apply
vagrant@ubuntu:~$ ip a show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
    link/ether 08:00:27:df:a2:e4 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.100.1/24 brd 192.168.100.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fd00:1337:cafe:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fedf:a2e4/64 scope link
```

```
        valid_lft forever preferred_lft forever
```

On Fedora:

```
[vagrant@fedora ~]$ ip a show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
    link/ether 08:00:27:7e:c1:07 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet6 fe80::e3fd:c3c5:a67d:e676/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[vagrant@fedora ~]$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  38615132-53d4-35ff-8df0-3b92d4588781  ethernet  eth0
lo                  5db8b605-105e-4625-b83a-bfd1f2417042  loopback  lo
enp0s3              76a80ae8-22e3-4baf-ad81-76bbbd334bf8  ethernet  --
Wired connection 2  32bfc162-a112-3607-b528-7b4cabf39cfe  ethernet  --
[vagrant@fedora ~]$ sudo nmcli connection modify 'Wired connection 2' ipv4.method manua
[vagrant@fedora ~]$ sudo nmcli connection up 'Wired connection 2'
[vagrant@fedora ~]$ ip a show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
    link/ether 08:00:27:7e:c1:07 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
   inet 192.168.100.2/24 brd 192.168.100.255 scope global noprefixroute eth1
        valid_lft forever preferred_lft forever
    inet6 fd00:1337:cafe:1::2/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::e3fd:c3c5:a67d:e676/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

4. Check if the VMs can ping each other. If not, troubleshoot the issue and fix it.

   On Ubuntu:

```
vagrant@ubuntu:~$ ping -c 1 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=1.88 ms

--- 192.168.100.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.881/1.881/1.881/0.000 ms
vagrant@ubuntu:~$ ping6 -c 1 fd00:1337:cafe:1::2
PING fd00:1337:cafe:1::2(fd00:1337:cafe:1::2) 56 data bytes
64 bytes from fd00:1337:cafe:1::2: icmp_seq=1 ttl=64 time=1.14 ms

--- fd00:1337:cafe:1::2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.142/1.142/1.142/0.000 ms
```

   On Fedora:

```
[vagrant@fedora ~]$ ping -c1 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=3.06 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 1ms
[vagrant@fedora ~]$ ping6 -c1 fd00:1337:cafe:1::1
PING fd00:1337:cafe:1::1(fd00:1337:cafe:1::1) 56 data bytes
64 bytes from fd00:1337:cafe:1::1: icmp_seq=1 ttl=64 time=3.95 ms

--- fd00:1337:cafe:1::1 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.951/3.951/3.951/0.000 ms
```

Both VMs can ping each other, so the network is working as expected.

5. Check the ARP cache on each VM. Is there a difference with the previous time you did this?

```
vagrant@ubuntu:~$ ip n
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
192.168.100.2 dev eth1 lladdr 08:00:27:7e:c1:07 STALE
fe80::e3fd:c3c5:a67d:e676 dev eth1 lladdr 08:00:27:7e:c1:07 DELAY
fd00:1337:cafe:1::2 dev eth1 lladdr 08:00:27:7e:c1:07 REACHABLE
```

```
[vagrant@fedora ~]$ ip n
192.168.100.1 dev eth1 lladdr 08:00:27:df:a2:e4 STALE
10.0.2.3 dev eth0 lladdr 52:54:00:12:35:03 STALE
10.0.2.2 dev eth0 lladdr 52:54:00:12:35:02 REACHABLE
fe80::a00:27ff:fedf:a2e4 dev eth1 lladdr 08:00:27:df:a2:e4 REACHABLE
fd00:1337:cafe:1::1 dev eth1 lladdr 08:00:27:df:a2:e4 REACHABLE
```

On both hosts, the ARP cache now contains an entry for the other host's `eth1` interface. This is expected, as we have just communicated with the other host. IPv6 addresses are also present in the overview, even though IPv6 uses the Neighbor Discovery Protocol (NDP) instead of ARP.

19. introduction to dhcp

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

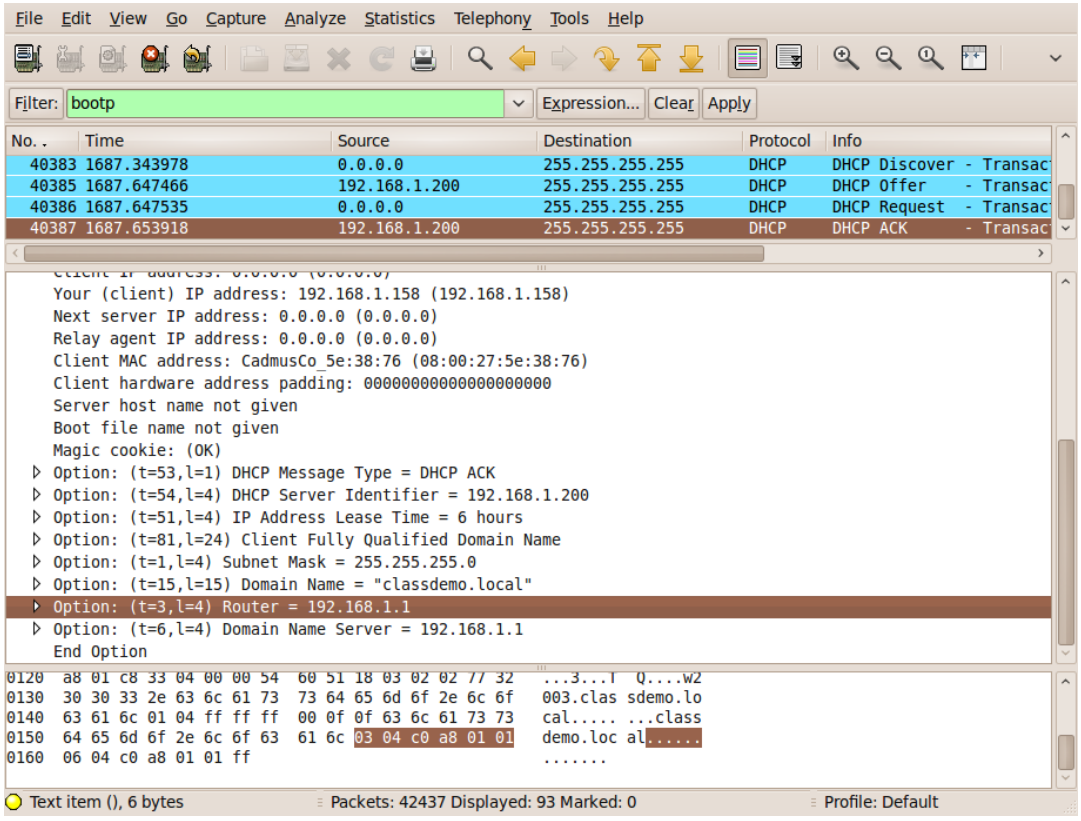
Dynamic Host Configuration Protocol (or short `dhcp`) is a standard tcp/ip protocol that distributes ip configurations to clients. `dhcp` is defined in `rfc 2131` (before that it was defined as an update to `bootp` in rfc 1531/1541.

The alternative to `dhcp` is manually entering the ip configuration on each client computer.

19.1. four broadcasts

`dhcp` works with layer 2 broadcasts. A dhcp client that starts, will send a `dhcp discover` on the network. All `dhcp servers` (that have a lease available) will respond with a `dhcp offer`. The client will choose one of those offers and will send a `dhcp request` containing the chosen offer. The `dhcp server` usually responds with a `dhcp ack`(knowledge).

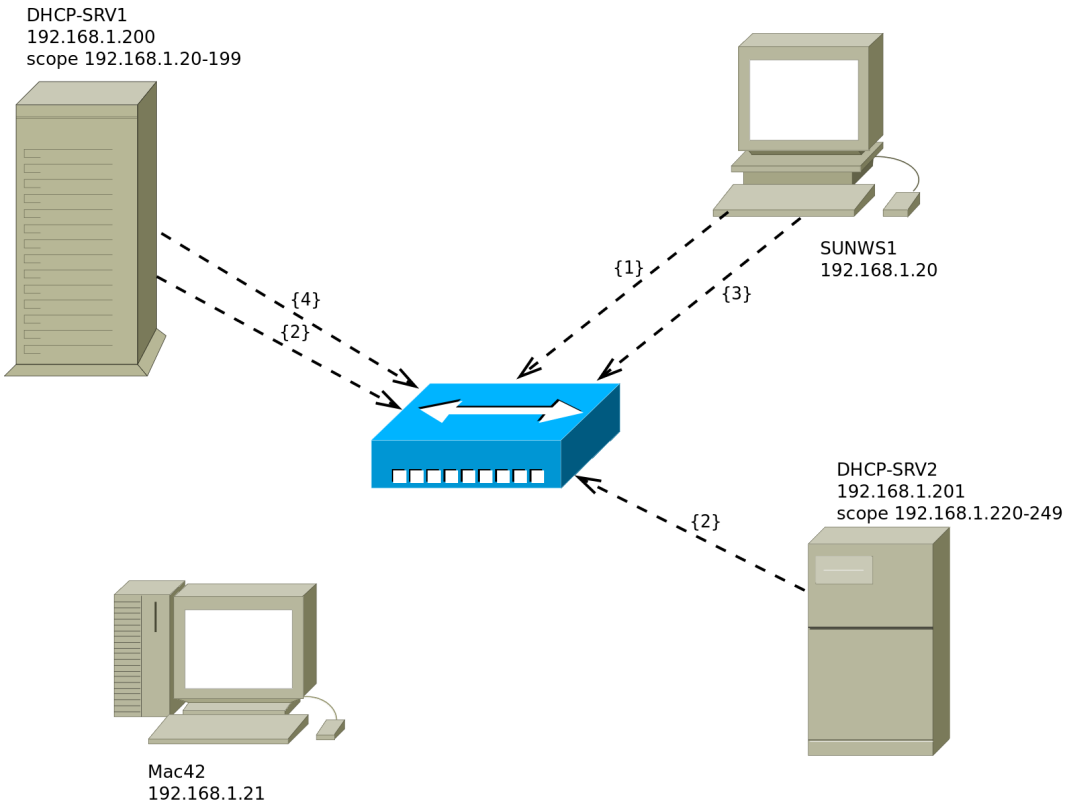
In wireshark it looks like this.



When this procedure is finished, then the client is allowed to use that ip-configuration until the end of its lease time.

19.2. picturing dhcp

Here we have a small network with two `dhcp servers` named DHCP-SRV1 and DHCP-SRV2 and two clients (SunWS1 and Mac42). All computers are connected by a hub or switch (pictured in the middle). All four computers have a cable to the hub (cables not pictured).



1. The client SunWS1 sends a `dhcp discover` on the network. All computers receive this broadcast.

2. Both `dhcp servers` answer with a `dhcp offer`. DHCP-SRV1 is a `dedicated dhcp server` and is faster in sending a `dhcp offer` than DHCP-SRV2 (who happens to also be a file server).

3. The client chooses the offer from DHCP-SRV1 and sends a `dhcp request` on the network.

4. DHCP-SRV1 answers with a `dhcp ack` (short for acknowledge).

All four broadcasts (or five when you count both offers) can be layer 2 ethernet broadcast to mac address `ff:ff:ff:ff:ff:ff` and a layer 3 ip broadcast to 255.255.255.255.

The same story can be read in `rfc 2131`.

19.3. installing a dhcp server

dhcp server for Debian/Mint

```
debian10:~# aptitude install dhcp3-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
```

```
The following NEW packages will be installed:
  dhcp3-server
```

You get a configuration file with many examples.

```
debian10:~# ls -l /etc/dhcp3/dhcpd.conf
-rw-r--r-- 1 root root 3551 2011-04-10 21:23 /etc/dhcp3/dhcpd.conf
```

19.4. dhcp server for RHEL/CentOS

Installing is easy with `yum`.

```
[root@linux ~]# yum install dhcp
Loaded plugins: product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package dhcp.x86_64 12:4.2.5-36.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package    Arch         Version                 Repository                Size
================================================================================
Installing:
 dhcp       x86_64       12:4.2.5-36.el7         rhel-7-server-rpms       510 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 510 k
Installed size: 1.4 M
Is this ok [y/d/N]: y
Downloading packages:
dhcp-4.2.5-36.el7.x86_64.rpm                               | 510 kB   00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 12:dhcp-4.2.5-36.el7.x86_64                                 1/1
  Verifying  : 12:dhcp-4.2.5-36.el7.x86_64                                 1/1

Installed:
  dhcp.x86_64 12:4.2.5-36.el7

Complete!
[root@linux ~]#
```

After installing we get a `/etc/dhcp/dhcpd.conf` that points us to an example file named `dhcpd.conf.sample`.

```
[root@linux ~]# cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
```

```
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
[root@linux ~]#
```

So we copy the sample and adjust it for our real situation. We name the copy `/etc/dhcp/dhcpd.conf`.

```
[root@linux ~]# cp /usr/share/doc/dhcp-4.2.5/dhcpd.conf.example /etc/dhcp/dhcp\
d.conf
[root@linux ~]# vi /etc/dhcp/dhcpd.conf
[root@linux ~]# cat /etc/dhcp/dhcpd.conf
option domain-name "linux-training.be";
option domain-name-servers 10.42.42.42;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 10.42.0.0 netmask 255.255.0.0 {
  range 10.42.200.11 10.42.200.120;
  option routers 10.42.200.1;
}
[root@linux ~]#
```

The 'routers' option is valid for the subnet alone, whereas the 'domain-name' option is global (for all subnets).

Time to start the server. Remember to use `systemctl start dhcpd` on RHEL7/CentOS8 and `service dhcpd start` on previous versions of RHEL/CentOS.

```
[root@linux ~]# systemctl start dhcpd
[root@linux ~]#
```

19.5. client reservations

You can reserve an ip configuration for a client using the mac address.

```
host pc42 {
hardware ethernet 11:22:33:44:55:66;
fixed-address 192.168.42.42;
}
```

You can add individual options to this reservation.

```
host pc42 {
hardware ethernet 11:22:33:44:55:66;
fixed-address 192.168.42.42;
option domain-name "linux-training.be";
option routers 192.168.42.1;
}
```

19.6. example config files

Below you see several sections of `/etc/dhcp/dhcpd.conf` on a `Debian 6` server.

```
# NetSec Antwerp Network
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.20 192.168.1.199;
 option domain-name-servers ns1.netsec.local;
 option domain-name "netsec.local";
 option routers 192.168.1.1;
 option broadcast-address 192.168.1.255;
 default-lease-time 7200;
 max-lease-time 7200;
}
```

Above the general configuration for the network, with a pool of 180 addresses.

Below two client reservations:

```
#
# laptops
#

host mac {
  hardware ethernet 00:26:bb:xx:xx:xx;
  fixed-address mac.netsec.local;
}

host vmac {
  hardware ethernet 8c:7b:9d:xx:xx:xx;
  fixed-address vmac.netsec.local;
}
```

19.7. older example config files

For dhcpd.conf on Fedora with dynamic updates for a DNS domain.

```
[root@fedora14 ~]# cat /etc/dhcp/dhcpd.conf
authoritative;
include "/etc/rndc.key";

log-facility local6;

server-identifier       fedora14;
ddns-domainname     "office.linux-training.be";
ddns-update-style   interim;
ddns-updates        on;
update-static-leases    on;

option domain-name  "office.linux-training.be";
option domain-name-servers  192.168.42.100;
option ip-forwarding    off;

default-lease-time  1800;
max-lease-time      3600;
```

```
zone office.linux-training.be {
    primary 192.168.42.100;
}

subnet 192.168.4.0 netmask 255.255.255.0 {
    range 192.168.4.24 192.168.4.40;
}
```

Allowing any updates in the zone database (part of the named.conf configuration)

```
zone "office.linux-training.be" {
    type master;
    file "/var/named/db.office.linux-training.be";
    allow-transfer { any; };
    allow-update { any; };
};
```

Allowing secure key updates in the zone database (part of the named.conf configuration)

```
zone "office.linux-training.be" {
    type master;
    file "/var/named/db.office.linux-training.be";
    allow-transfer { any; };
    allow-update { key mykey; };
};
```

Sample key file contents:

```
[root@fedora14 ~]# cat /etc/rndc.key
key "rndc-key" {
    algorithm hmac-md5;
    secret "4Ykd58uIeUr3Ve6ad1qTfQ==";
};
```

Generate your own keys with `dnssec-keygen`.

How to include a key in a config file:

```
include "/etc/bind/rndc.key";
```

Also make sure that `bind` can write to your db.zone file (using chmod/chown). For Ubuntu this can be in /etc/bind, for Fedora in /var/named.

19.8. advanced dhcp

19.8.1. 80/20 rule

DHCP servers should not be a single point of failure. Let us discuss redundant dhcp server setups.

19.8.2. relay agent

To avoid having to place a dhcp server on every segment, we can use `dhcp relay agents`.

19.8.3. rogue dhcp servers

Rogue dhcp servers are a problem without a solution. For example accidental connection of a (believed to be simple) hub/switch to a network with an internal dhcp server.

19.8.4. dhcp and ddns

DHCP can dynamically update DNS when it configures a client computer. DDNS can be used with or without secure keys.

When set up properly records can be added automaticall to the zone file:

```
root@fedora14~# tail -2 /var/named/db.office.linux-training.be
ubu1010srv          A     192.168.42.151
                    TXT   "00dfbb15e144a273c3cf2d6ae933885782"
```

19.9. Practice: dhcp

1. Make sure you have a unique fixed ip address for your DNS and DHCP server (easier on the same machine).

2. Install DHCP and browse the explanation in the default configuration file /etc/dhcp/dhcpd.conf or /etc/dhcp3/dhcpd.conf.

3. Decide on a valid scope and activate it.

4. Test with a client that your DHCP server works.

5. Use wireshark to capture the four broadcasts when a client receives an ip (for the first time).

6. Use wireshark to capture a DHCPNAK and a DHCPrelease.

7. Reserve a configuration for a particular client (using mac address).

8. Configure your DHCP/DNS server(s) with a proper hostname and domainname (/etc/hosts, /etc/hostname, /etc/sysconfig/network on Fedora/RHEL, /etc/resolv.conf ...). You may need to disable NetworkManager on *buntu-desktops.

9. Make sure your DNS server still works, and is master over (at least) one domain.

There are several ways to do steps 10-11-12. Google is your friend in exploring DDNS with keys, with key-files or without keys.

10. Configure your DNS server to allow dynamic updates from your DHCP server.

11. Configure your DHCP server to send dynamic updates to your DNS server.

12. Test the working of Dynamic DNS.

Part V.

Webserver; scripting 102

20. apache web server

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Hans Roes, https://github.com/Blokker-1999/, Alex M. Schapelle, https://github.com/zero-pytagoras/)

In this chapter we learn how to setup a web server with the `apache` software.

According to NetCraft (http://news.netcraft.com/archives/web_server_survey.html) about seventy percent of all web servers are running on Apache. The name is derived from a `patchy` web server, because of all the patches people wrote for the NCSA httpd server.

Later chapters will expand this web server into a LAMP stack (Linux, Apache, Mysql, Perl/PHP/Python).

20.1. introduction to apache

20.1.1. installing on Debian

This screenshot shows that there is no `apache` server installed, nor does the `/var/www` directory exist.

```
root@linux:~# ls -l /var/www
ls: cannot access /var/www: No such file or directory
root@linux:~# dpkg -l | grep apache
```

To install `apache` on Debian:

```
root@linux:~# aptitude install apache2
The following NEW packages will be installed:
  apache2 apache2-mpm-worker{a} apache2-utils{a} apache2.2-bin{a} apache2.2-
com\
mon{a}  libapr1{a}  libaprutil1{a}  libaprutil1-dbd-sqlite3{a}  libaprutil1-
ldap{a}\
 ssl-cert{a}
0 packages upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,487 kB of archives. After unpacking 5,673 kB will be used.
Do you want to continue? [Y/n/?]
```

After installation, the same two commands as above will yield a different result:

```
root@linux:~# ls -l /var/www
total 4
-rw-r--r-- 1 root root 177 Apr 29 11:55 index.html
root@linux:~# dpkg -l | grep apache | tr -s ' '
ii apache2 2.2.22-13+deb7u1 amd64 Apache HTTP Server metapackage
ii apache2-mpm-worker 2.2.22-13+deb7u1 amd64 Apache HTTP Server - high speed th\
readed model
ii apache2-utils 2.2.22-13+deb7u1 amd64 utility programs for webservers
ii apache2.2-bin 2.2.22-13+deb7u1 amd64 Apache HTTP Server common binary files
ii apache2.2-common 2.2.22-13+deb7u1 amd64 Apache HTTP Server common files
```

20.1.2. installing on RHEL/CentOS

Note that Red Hat derived distributions use `httpd` as package and process name instead of `apache`.

To verify whether `apache` is installed in CentOS/RHEL:

```
[root@linux ~]# rpm -q httpd
package httpd is not installed
[root@linux ~]# ls -l /var/www
ls: cannot access /var/www: No such file or directory
```

To install apache on CentOS:

```
[root@linux ~]# yum install httpd
```

After running the `yum install httpd` command, the Centos 6.5 server has apache installed and the `/var/www` directory exists.

```
[root@linux ~]# rpm -q httpd
httpd-2.2.15-30.el6.centos.x86_64
[root@linux ~]# ls -l /var/www
total 16
drwxr-xr-x. 2 root root 4096 Apr  3 23:57 cgi-bin
drwxr-xr-x. 3 root root 4096 May  6 13:08 error
drwxr-xr-x. 2 root root 4096 Apr  3 23:57 html
drwxr-xr-x. 3 root root 4096 May  6 13:08 icons
[root@linux ~]#
```

20.1.3. running apache on Debian

This is how you start `apache2` on Debian.

```
root@linux:~# service apache2 status
Apache2 is NOT running.
root@linux:~# service apache2 start
Starting web server: apache2apache2: Could not reliably determine the server's \
fully qualified domain name, using 127.0.1.1 for ServerName
.
```

To verify, run the `service apache2 status` command again or use `ps`.

```
root@linux:~# service apache2 status
Apache2 is running (pid 3680).
root@linux:~# ps -C apache2
  PID TTY          TIME CMD
 3680 ?        00:00:00 apache2
 3683 ?        00:00:00 apache2
 3684 ?        00:00:00 apache2
 3685 ?        00:00:00 apache2
root@linux:~#
```

Or use `wget` and `file` to verify that your web server serves an html document.

```
root@linux:~# wget 127.0.0.1
--2014-05-06 13:27:02--  http://127.0.0.1/
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177 [text/html]
Saving to: `index.html'

100%[======================================================>]  177         --.-
K/s   in 0s

2014-05-06 13:27:02 (15.8 MB/s) - `index.html' saved [177/177]

root@linux:~# file index.html
index.html: HTML document, ASCII text
root@linux:~#
```

Or verify that apache is running by opening a web browser, and browse to the ip-address of your server. An Apache test page should be shown.

You can do the following to quickly avoid the 'could not reliably determine the fqdn' message when restarting apache.

```
root@linux:~# echo ServerName debian10 >> /etc/apache2/apache2.conf
root@linux:~# service apache2 restart
Restarting web server: apache2 ... waiting .
root@linux:~#
```

20.1.4. running apache on CentOS

Starting the `httpd` on RHEL/CentOS is done with the `service` command.

```
[root@linux ~]# service httpd status
httpd is stopped
[root@linux ~]# service httpd start
Starting httpd: httpd: Could not reliably determine the server's fully qualifie\
d domain name, using 127.0.0.1 for ServerName
                                                           [  OK  ]
[root@linux ~]#
```

To verify that `apache` is running, use `ps` or issue the `service httpd status` command again.

```
[root@linux ~]# service httpd status
httpd (pid  2410) is running...
[root@linux ~]# ps -C httpd
  PID TTY          TIME CMD
 2410 ?        00:00:00 httpd
 2412 ?        00:00:00 httpd
 2413 ?        00:00:00 httpd
 2414 ?        00:00:00 httpd
 2415 ?        00:00:00 httpd
 2416 ?        00:00:00 httpd
 2417 ?        00:00:00 httpd
 2418 ?        00:00:00 httpd
 2419 ?        00:00:00 httpd
[root@linux ~]#
```

To prevent the 'Could not reliably determine the fqdn' message, issue the following command.

```
[root@linux ~]# echo ServerName Centos65 >> /etc/httpd/conf/httpd.conf
[root@linux ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@linux ~]#
```

20.1.5. index file on CentOS

CentOS does not provide a standard index.html or index.php file. A simple `wget` gives an error.

```
[root@linux ~]# wget 127.0.0.1
--2014-05-06 15:10:22--  http://127.0.0.1/
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-05-06 15:10:22 ERROR 403: Forbidden.
```

Instead when visiting the ip-address of your server in a web browser you get a `noindex.html` page. You can verify this using `wget`.

```
[root@linux ~]# wget http://127.0.0.1/error/noindex.html
--2014-05-06 15:16:05--  http://127.0.0.1/error/noindex.html
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5039 (4.9K) [text/html]
Saving to: “noindex.html”

100%[======================================>] 5,039     --.-K/s  in 0s

2014-05-06 15:16:05 (289 MB/s) - “noindex.html” saved [5039/5039]

[root@linux ~]# file noindex.html
noindex.html: HTML document text
[root@linux ~]#
```

Any custom `index.html` file in `/var/www/html` will immediately serve as an index for this web server.

```
[root@linux ~]# echo 'Welcome to my website' > /var/www/html/index.html
[root@linux ~]# wget http://127.0.0.1
--2014-05-06 15:19:16--  http://127.0.0.1/
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22 [text/html]
Saving to: “index.html”

100%[======================================>] 22        --.-K/s  in 0s

2014-05-06 15:19:16 (1.95 MB/s) - “index.html” saved [22/22]

[root@linux ~]# cat index.html
Welcome to my website
```

20.1.6. default website

Changing the default website of a freshly installed apache web server is easy. All you need to do is create (or change) an index.html file in the DocumentRoot directory.

To locate the DocumentRoot directory on Debian:

```
root@linux:~# grep DocumentRoot /etc/apache2/sites-available/default
        DocumentRoot /var/www
```

This means that `/var/www/index.html` is the default web site.

```
root@linux:~# cat /var/www/index.html
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
root@linux:~#
```

This screenshot shows how to locate the `DocumentRoot` directory on RHEL/CentOS.

```
[root@linux ~]# grep ^DocumentRoot /etc/httpd/conf/httpd.conf
DocumentRoot "/var/www/html"
```

RHEL/CentOS have no default web page (only the noindex.html error page mentioned before). But an `index.html` file created in `/var/www/html/` will automatically be used as default page.

```
[root@linux ~]# echo '<html><head><title>Default website</title></head><body\
><p>A new web page</p></body></html>' > /var/www/html/index.html
[root@linux ~]# cat /var/www/html/index.html
<html><head><title>Default website</title></head><body><p>A new web page</p></b\
ody></html>
[root@linux ~]#
```

20.1.7. apache configuration

There are many similarities, but also a couple of differences when configuring `apache` on Debian or on CentOS. Both Linux families will get their own chapters with examples.

All configuration on RHEL/CentOS is done in `/etc/httpd`.

```
[root@linux ~]# ls -l /etc/httpd/
total 8
drwxr-xr-x. 2 root root 4096 May  6 13:08 conf
drwxr-xr-x. 2 root root 4096 May  6 13:08 conf.d
lrwxrwxrwx. 1 root root   19 May  6 13:08 logs -> ../../var/log/httpd
lrwxrwxrwx. 1 root root   29 May  6 13:08 modules -> ../../usr/lib64/httpd/modu\
les
lrwxrwxrwx. 1 root root   19 May  6 13:08 run -> ../../var/run/httpd
[root@linux ~]#
```

Debian (and ubuntu/mint/...) use `/etc/apache2`.

```
root@linux:~# ls -l /etc/apache2/
total 72
-rw-r--r-- 1 root root  9659 May  6 14:23 apache2.conf
drwxr-xr-x 2 root root  4096 May  6 13:19 conf.d
-rw-r--r-- 1 root root  1465 Jan 31 18:35 envvars
-rw-r--r-- 1 root root 31063 Jul 20  2013 magic
drwxr-xr-x 2 root root  4096 May  6 13:19 mods-available
drwxr-xr-x 2 root root  4096 May  6 13:19 mods-enabled
-rw-r--r-- 1 root root   750 Jan 26 12:13 ports.conf
drwxr-xr-x 2 root root  4096 May  6 13:19 sites-available
drwxr-xr-x 2 root root  4096 May  6 13:19 sites-enabled
root@linux:~#
```

20.2. port virtual hosts on Debian

20.2.1. default virtual host

Debian has a virtualhost configuration file for its default website in `/etc/apache2/sites-available/default`.

```
root@linux:~# head -2 /etc/apache2/sites-available/default
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
```

20.2.2. three extra virtual hosts

In this scenario we create three additional websites for three customers that share a clubhouse and want to jointly hire you. They are a model train club named `Choo Choo`, a chess club named `Chess Club 42` and a hackerspace named `hunter2`.

One way to put three websites on one web server, is to put each website on a different port. This screenshot shows three newly created `virtual hosts`, one for each customer.

```
root@linux:~# vi /etc/apache2/sites-available/choochoo
root@linux:~# cat /etc/apache2/sites-available/choochoo
<VirtualHost *:7000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/choochoo
</VirtualHost>
root@linux:~# vi /etc/apache2/sites-available/chessclub42
root@linux:~# cat /etc/apache2/sites-available/chessclub42
<VirtualHost *:8000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/chessclub42
</VirtualHost>
root@linux:~# vi /etc/apache2/sites-available/hunter2
root@linux:~# cat /etc/apache2/sites-available/hunter2
<VirtualHost *:9000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/hunter2
</VirtualHost>
```

Notice the different port numbers 7000, 8000 and 9000. Notice also that we specified a unique `DocumentRoot` for each website.

Are you using `Ubuntu` or `Mint`, then these configfiles need to end in `.conf`.

20.2.3. three extra ports

We need to enable these three ports on apache in the `ports.conf` file. Open this file with `vi` and add three lines to `listen` on three extra ports.

```
root@linux:~# vi /etc/apache2/ports.conf
```

Verify with `grep` that the `Listen` directives are added correctly.

```
root@linux:~# grep ^Listen /etc/apache2/ports.conf
Listen 80
Listen 7000
Listen 8000
Listen 9000
```

20.2.4. three extra websites

Next we need to create three `DocumentRoot` directories.

```
root@linux:~# mkdir /var/www/choochoo
root@linux:~# mkdir /var/www/chessclub42
root@linux:~# mkdir /var/www/hunter2
```

And we have to put some really simple website in those directories.

```
root@linux:~# echo 'Choo Choo model train Choo Choo' > /var/www/choochoo/inde\
x.html
root@linux:~# echo 'Welcome to chess club 42' > /var/www/chessclub42/index.ht\
ml
root@linux:~# echo 'HaCkInG iS fUn At HuNtEr2' > /var/www/hunter2/index.html
```

20.2.5. enabling extra websites

The last step is to enable the websites with the `a2ensite` command. This command will create links in `sites-enabled`.

The links are not there yet...

```
root@linux:~# cd /etc/apache2/
root@linux:/etc/apache2# ls sites-available/
chessclub42  choochoo  default  default-ssl  hunter2
root@linux:/etc/apache2# ls sites-enabled/
000-default
```

So we run the `a2ensite` command for all websites.

```
root@linux:/etc/apache2# a2ensite choochoo
Enabling site choochoo.
To activate the new configuration, you need to run:
  service apache2 reload
root@linux:/etc/apache2# a2ensite chessclub42
Enabling site chessclub42.
To activate the new configuration, you need to run:
  service apache2 reload
```

```
root@linux:/etc/apache2# a2ensite hunter2
Enabling site hunter2.
To activate the new configuration, you need to run:
  service apache2 reload
```

The links are created, so we can tell apache.

```
root@linux:/etc/apache2# ls sites-enabled/
000-default  chessclub42  choochoo  hunter2
root@linux:/etc/apache2# service apache2 reload
Reloading web server config: apache2.
root@linux:/etc/apache2#
```

20.2.6. testing the three websites

Testing the model train club named Choo Choo on port 7000.

```
root@linux:/etc/apache2# wget 127.0.0.1:7000
--2014-05-06 21:16:03--  http://127.0.0.1:7000/
Connecting to 127.0.0.1:7000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32 [text/html]
Saving to: `index.html'

100%[======================================>] 32          --.-K/s   in 0s

2014-05-06 21:16:03 (2.92 MB/s) - `index.html' saved [32/32]

root@linux:/etc/apache2# cat index.html
Choo Choo model train Choo Choo
```

Testing the chess club named Chess Club 42 on port 8000.

```
root@linux:/etc/apache2# wget 127.0.0.1:8000
--2014-05-06 21:16:20--  http://127.0.0.1:8000/
Connecting to 127.0.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25 [text/html]
Saving to: `index.html.1'

100%[======================================>] 25          --.-K/s   in 0s

2014-05-06 21:16:20 (2.16 MB/s) - `index.html.1' saved [25/25]

root@linux:/etc/apache2# cat index.html.1
Welcome to chess club 42
```

Testing the hacker club named hunter2 on port 9000.

```
root@linux:/etc/apache2# wget 127.0.0.1:9000
--2014-05-06 21:16:30--  http://127.0.0.1:9000/
Connecting to 127.0.0.1:9000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26 [text/html]
Saving to: `index.html.2'
```

```
100%[======================================>] 26          --.-K/s   in 0s

2014-05-06 21:16:30 (2.01 MB/s) - `index.html.2' saved [26/26]

root@linux:/etc/apache2# cat index.html.2
HaCkInG iS fUn At HuNtEr2
```

Cleaning up the temporary files.

```
root@linux:/etc/apache2# rm index.html index.html.1 index.html.2
```

Try testing from another computer using the ip-address of your server.

20.3. named virtual hosts on Debian

20.3.1. named virtual hosts

The chess club and the model train club find the port numbers too hard to remember. They would prefere to have their website accessible by name.

We continue work on the same server that has three websites on three ports. We need to make sure those websites are accesible using the names `choochoo.local`, `chessclub42.local` and `hunter2.local`.

We start by creating three new virtualhosts.

```
root@linux:/etc/apache2/sites-available# vi choochoo.local
root@linux:/etc/apache2/sites-available# vi chessclub42.local
root@linux:/etc/apache2/sites-available# vi hunter2.local
root@linux:/etc/apache2/sites-available# cat choochoo.local
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName choochoo.local
        DocumentRoot /var/www/choochoo
</VirtualHost>
root@linux:/etc/apache2/sites-available# cat chessclub42.local
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName chessclub42.local
        DocumentRoot /var/www/chessclub42
</VirtualHost>
root@linux:/etc/apache2/sites-available# cat hunter2.local
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName hunter2.local
        DocumentRoot /var/www/hunter2
</VirtualHost>
root@linux:/etc/apache2/sites-available#
```

Notice that they all listen on `port 80` and have an extra `ServerName` directive.

20.3.2. name resolution

We need some way to resolve names. This can be done with DNS, which is discussed in another chapter. For this demo it is also possible to quickly add the three names to the `/etc/hosts` file.

```
root@linux:/etc/apache2/sites-available# grep ^192 /etc/hosts
192.168.42.50 choochoo.local
192.168.42.50 chessclub42.local
192.168.42.50 hunter2.local
```

Note that you may have another ip address...

20.3.3. enabling virtual hosts

Next we enable them with `a2ensite`.

```
root@linux:/etc/apache2/sites-available# a2ensite choochoo.local
Enabling site choochoo.local.
To activate the new configuration, you need to run:
  service apache2 reload
root@linux:/etc/apache2/sites-available# a2ensite chessclub42.local
Enabling site chessclub42.local.
To activate the new configuration, you need to run:
  service apache2 reload
root@linux:/etc/apache2/sites-available# a2ensite hunter2.local
Enabling site hunter2.local.
To activate the new configuration, you need to run:
  service apache2 reload
```

20.3.4. reload and verify

After a `service apache2 reload` the websites should be available by name.

```
root@linux:/etc/apache2/sites-available# service apache2 reload
Reloading web server config: apache2.
root@linux:/etc/apache2/sites-available# wget chessclub42.local
--2014-05-06 21:37:13--  http://chessclub42.local/
Resolving chessclub42.local (chessclub42.local)... 192.168.42.50
Connecting to chessclub42.local (chessclub42.local)|192.168.42.50|:80... conne\
cted.
HTTP request sent, awaiting response... 200 OK
Length: 25 [text/html]
Saving to: `index.html'

100%[======================================>] 25          --.-K/s   in 0s

2014-05-06 21:37:13 (2.06 MB/s) - `index.html' saved [25/25]

root@linux:/etc/apache2/sites-available# cat index.html
Welcome to chess club 42
```

20.4. password protected website on Debian

You can secure files and directories in your website with a `.htaccess` file that refers to a `.htpasswd` file. The `htpasswd` command can create a `.htpasswd` file that contains a userid and an (encrypted) password.

This screenshot creates a user and password for the hacker named `cliff` and uses the `-c` flag to create the `.htpasswd` file.

```
root@linux:~# htpasswd -c /var/www/.htpasswd cliff
New password:
Re-type new password:
Adding password for user cliff
root@linux:~# cat /var/www/.htpasswd
cliff:$apr1$vujll0KL$./SZ4w9q0swhX93pQ0PVp.
```

Hacker `rob` also wants access, this screenshot shows how to add a second user and password to `.htpasswd`.

```
root@linux:~# htpasswd /var/www/.htpasswd rob
New password:
Re-type new password:
Adding password for user rob
root@linux:~# cat /var/www/.htpasswd
cliff:$apr1$vujll0KL$./SZ4w9q0swhX93pQ0PVp.
rob:$apr1$HNln1FFt$nRlpF0H.IW11/1DRq4lQo0
```

Both Cliff and Rob chose the same password (hunter2), but that is not visible in the `.htpasswd` file because of the different salts.

Next we need to create a `.htaccess` file in the `DocumentRoot` of the website we want to protect. This screenshot shows an example.

```
root@linux:~# cd /var/www/hunter2/
root@linux:/var/www/hunter2# cat .htaccess
AuthUserFile /var/www/.htpasswd
AuthName "Members only!"
AuthType Basic
require valid-user
```

Note that we are protecting the website on `port 9000` that we created earlier.

And because we put the website for the Hackerspace named hunter2 in a subdirectory of the default website, we will need to adjust the `AllowOvveride` parameter in `/etc/apache2/sites-available/default` as this screenshot shows (with line numbers on debian10, your may vary).

```
9         <Directory /var/www/>
10                Options Indexes FollowSymLinks MultiViews
11                AllowOverride Authconfig
12                Order allow,deny
13                allow from all
14         </Directory
```

Now restart the apache2 server and test that it works!

20.5. port virtual hosts on CentOS

20.5.1. default virtual host

Unlike Debian, CentOS has no virtualHost configuration file for its default website. Instead the default configuration will throw a standard error page when no index file can be found in the default location (/var/www/html).

20.5.2. three extra virtual hosts

In this scenario we create three additional websites for three customers that share a clubhouse and want to jointly hire you. They are a model train club named `Choo Choo`, a chess club named `Chess Club 42` and a hackerspace named `hunter2`.

One way to put three websites on one web server, is to put each website on a different port. This screenshot shows three newly created `virtual hosts`, one for each customer.

```
[root@CentOS65 ~]# vi /etc/httpd/conf.d/choochoo.conf
[root@CentOS65 ~]# cat /etc/httpd/conf.d/choochoo.conf
<VirtualHost *:7000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/choochoo
</VirtualHost>
[root@CentOS65 ~]# vi /etc/httpd/conf.d/chessclub42.conf
[root@CentOS65 ~]# cat /etc/httpd/conf.d/chessclub42.conf
<VirtualHost *:8000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/chessclub42
</VirtualHost>
[root@CentOS65 ~]# vi /etc/httpd/conf.d/hunter2.conf
[root@CentOS65 ~]# cat /etc/httpd/conf.d/hunter2.conf
<VirtualHost *:9000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/hunter2
</VirtualHost>
```

Notice the different port numbers 7000, 8000 and 9000. Notice also that we specified a unique `DocumentRoot` for each website.

20.5.3. three extra ports

We need to enable these three ports on apache in the `httpd.conf` file.

```
[root@CentOS65 ~]# vi /etc/httpd/conf/httpd.conf
root@linux:~# grep ^Listen /etc/httpd/conf/httpd.conf
Listen 80
Listen 7000
Listen 8000
Listen 9000
```

20.5.4. SELinux guards our ports

If we try to restart our server, we will notice the following error:

```
[root@CentOS65 ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:
      (13)Permission denied: make_sock: could not bind to address 0.0.0.0:7000
no listening sockets available, shutting down
                                                           [FAILED]
```

This is due to SELinux reserving ports 7000 and 8000 for other uses. We need to tell SELinux we want to use these ports for http traffic

```
[root@CentOS65 ~]# semanage port -m -t http_port_t -p tcp 7000
[root@CentOS65 ~]# semanage port -m -t http_port_t -p tcp 8000
[root@CentOS65 ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

20.5.5. three extra websites

Next we need to create three `DocumentRoot` directories.

```
[root@CentOS65 ~]# mkdir /var/www/html/choochoo
[root@CentOS65 ~]# mkdir /var/www/html/chessclub42
[root@CentOS65 ~]# mkdir /var/www/html/hunter2
```

And we have to put some really simple website in those directories.

```
[root@CentOS65 ~]# echo 'Choo Choo model train Choo Choo' > /var/www/html/chooc\
hoo/index.html
[root@CentOS65 ~]# echo 'Welcome to chess club 42' > /var/www/html/chessclub42/\
index.html
[root@CentOS65 ~]# echo 'HaCkInG iS fUn At HuNtEr2' > /var/www/html/hunter2/ind\
ex.html
```

20.5.6. enabling extra websites

The only way to enable or disable configurations in RHEL/CentOS is by renaming or moving the configuration files. Any file in /etc/httpd/conf.d ending on .conf will be loaded by Apache. To disable a site we can either rename the file or move it to another directory.

The files are created, so we can tell `apache`.

```
[root@CentOS65 ~]# ls /etc/httpd/conf.d/
chessclub42.conf  choochoo.conf  hunter2.conf  README  welcome.conf
[root@CentOS65 ~]# service httpd reload
Reloading httpd:
```

20.5.7. testing the three websites

Testing the model train club named `Choo Choo` on port 7000.

```
[root@CentOS65 ~]# wget 127.0.0.1:7000
--2014-05-11 11:59:36--  http://127.0.0.1:7000/
Connecting to 127.0.0.1:7000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32 [text/html]
Saving to: `index.html'

100%[======================================>] 32          --.-K/s   in 0s

2014-05-11 11:59:36 (4.47 MB/s) - `index.html' saved [32/32]

[root@CentOS65 ~]# cat index.html
Choo Choo model train Choo Choo
```

Testing the chess club named `Chess Club 42` on port 8000.

```
[root@CentOS65 ~]# wget 127.0.0.1:8000
--2014-05-11 12:01:30--  http://127.0.0.1:8000/
Connecting to 127.0.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25 [text/html]
Saving to: `index.html.1'

100%[======================================>] 25          --.-K/s   in 0s

2014-05-11 12:01:30 (4.25 MB/s) - `index.html.1' saved [25/25]

root@linux:/etc/apache2# cat index.html.1
Welcome to chess club 42
```

Testing the hacker club named `hunter2` on port 9000.

```
[root@CentOS65 ~]# wget 127.0.0.1:9000
--2014-05-11 12:02:37--  http://127.0.0.1:9000/
Connecting to 127.0.0.1:9000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26 [text/html]
Saving to: `index.html.2'

100%[======================================>] 26          --.-K/s   in 0s

2014-05-11 12:02:37 (4.49 MB/s) - `index.html.2' saved [26/26]

root@linux:/etc/apache2# cat index.html.2
HaCkInG iS fUn At HuNtEr2
```

Cleaning up the temporary files.

```
[root@CentOS65 ~]# rm index.html index.html.1 index.html.2
```

20.5.8. firewall rules

If we attempt to access the site from another machine however, we will not be able to view the website yet. The firewall is blocking incoming connections. We need to open these incoming ports first

```
[root@CentOS65 ~]# iptables -I INPUT -p tcp --dport 80 -j ACCEPT
[root@CentOS65 ~]# iptables -I INPUT -p tcp --dport 7000 -j ACCEPT
[root@CentOS65 ~]# iptables -I INPUT -p tcp --dport 8000 -j ACCEPT
[root@CentOS65 ~]# iptables -I INPUT -p tcp --dport 9000 -j ACCEPT
```

And if we want these rules to remain active after a reboot, we need to save them

```
[root@CentOS65 ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

20.6. named virtual hosts on CentOS

20.6.1. named virtual hosts

The chess club and the model train club find the port numbers too hard to remember. They would prefere to have their website accessible by name.

We continue work on the same server that has three websites on three ports. We need to make sure those websites are accesible using the names `choochoo.local`, `chessclub42.local` and `hunter2.local`.

First, we need to enable named virtual hosts in the configuration

```
[root@CentOS65 ~]# vi /etc/httpd/conf/httpd.conf
[root@CentOS65 ~]# grep ^NameVirtualHost /etc/httpd/conf/httpd.conf
NameVirtualHost *:80
[root@CentOS65 ~]#
```

Next we need to create three new virtualhosts.

```
[root@CentOS65 ~]# vi /etc/httpd/conf.d/choochoo.local.conf
[root@CentOS65 ~]# vi /etc/httpd/conf.d/chessclub42.local.conf
[root@CentOS65 ~]# vi /etc/httpd/conf.d/hunter2.local.conf
[root@CentOS65 ~]# cat /etc/httpd/conf.d/choochoo.local.conf
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName choochoo.local
        DocumentRoot /var/www/html/choochoo
</VirtualHost>
[root@CentOS65 ~]# cat /etc/httpd/conf.d/chessclub42.local.conf
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName chessclub42.local
        DocumentRoot /var/www/html/chessclub42
</VirtualHost>
[root@CentOS65 ~]# cat /etc/httpd/conf.d/hunter2.local.conf
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName hunter2.local
```

```
        DocumentRoot /var/www/html/hunter2
</VirtualHost>
[root@CentOS65 ~]#
```

Notice that they all listen on `port 80` and have an extra `ServerName` directive.

20.6.2. name resolution

We need some way to resolve names. This can be done with DNS, which is discussed in another chapter. For this demo it is also possible to quickly add the three names to the `/etc/hosts` file.

```
[root@CentOS65 ~]# grep ^192 /etc/hosts
192.168.1.225 choochoo.local
192.168.1.225 chessclub42.local
192.168.1.225 hunter2.local
```

Note that you may have another ip address...

20.6.3. reload and verify

After a `service httpd reload` the websites should be available by name.

```
[root@CentOS65 ~]# service httpd reload
Reloading httpd:
[root@CentOS65 ~]# wget chessclub42.local
--2014-05-25 16:59:14--  http://chessclub42.local/
Resolving chessclub42.local... 192.168.1.225
Connecting to chessclub42.local|192.168.1.225|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25 [text/html]
Saving to: âindex.htmlâ

100%[======================================>] 25          --.-K/s   in 0s

2014-05-25 16:59:15 (1014 KB/s) - `index.html' saved [25/25]

[root@CentOS65 ~]# cat index.html
Welcome to chess club 42
```

20.7. password protected website on CentOS

You can secure files and directories in your website with a `.htaccess` file that refers to a `.htpasswd` file. The `htpasswd` command can create a `.htpasswd` file that contains a userid and an (encrypted) password.

This screenshot creates a user and password for the hacker named `cliff` and uses the `-c` flag to create the `.htpasswd` file.

```
[root@CentOS65 ~]# htpasswd -c /var/www/.htpasswd cliff
New password:
Re-type new password:
Adding password for user cliff
[root@CentOS65 ~]# cat /var/www/.htpasswd
cliff:QNwTrymMLBctU
```

Hacker `rob` also wants access, this screenshot shows how to add a second user and password to `.htpasswd`.

```
[root@CentOS65 ~]# htpasswd /var/www/.htpasswd rob
New password:
Re-type new password:
Adding password for user rob
[root@CentOS65 ~]# cat /var/www/.htpasswd
cliff:QNwTrymMLBctU
rob:EC2vOCcrMXDoM
[root@CentOS65 ~]#
```

Both Cliff and Rob chose the same password (hunter2), but that is not visible in the `.htpasswd` file because of the different salts.

Next we need to create a `.htaccess` file in the `DocumentRoot` of the website we want to protect. This screenshot shows an example.

```
[root@CentOS65 ~]# cat /var/www/html/hunter2/.htaccess
AuthUserFile /var/www/.htpasswd
AuthName "Members only!"
AuthType Basic
require valid-user
```

Note that we are protecting the website on `port 9000` that we created earlier.

And because we put the website for the Hackerspace named hunter2 in a subdirectory of the default website, we will need to adjust the `AllowOvveride` parameter in `/etc/httpd/conf/httpd.conf` under the `<Directory "/var/www/html">` directive as this screenshot shows.

```
[root@CentOS65 ~]# vi /etc/httpd/conf/httpd.conf

<Directory "/var/www/html">

#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs/2.2/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
```

```
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride Authconfig

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all

</Directory>
```

Now restart the apache2 server and test that it works!

20.8. troubleshooting apache

When apache restarts, it will verify the syntax of files in the configuration folder `/etc/apache2` on debian or `/etc/httpd` on CentOS and it will tell you the name of the faulty file, the line number and an explanation of the error.

```
root@linux:~# service apache2 restart
apache2: Syntax error on line 268 of /etc/apache2/apache2.conf: Syntax error o\
n  line  1  of  /etc/apache2/sites-enabled/chessclub42:  /etc/apache2/sites-
enabled\
/chessclub42:4: <VirtualHost> was not closed.\n/etc/apache2/sites-enabled/ches\
sclub42:1: <VirtualHost> was not closed.
Action 'configtest' failed.
The Apache error log may have more information.
 failed!
```

Below you see the problem... a missing / before on line 4.

```
root@linux:~# cat /etc/apache2/sites-available/chessclub42
<VirtualHost *:8000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/chessclub42
<VirtualHost>
```

Let us force another error by renaming the directory of one of our websites:

```
root@linux:~# mv /var/www/choochoo/ /var/www/chooshoo
root@linux:~# !ser
service apache2 restart
Restarting web server: apache2Warning: DocumentRoot [/var/www/choochoo] does n\
ot exist
Warning: DocumentRoot [/var/www/choochoo] does not exist
  ... waiting Warning: DocumentRoot [/var/www/choochoo] does not exist
Warning: DocumentRoot [/var/www/choochoo] does not exist
.
```

As you can see, apache will tell you exactly what is wrong.

You can also troubleshoot by connecting to the website via a browser and then checking the apache log files in `/var/log/apache`.

20.9. virtual hosts example

Below is a sample virtual host configuration. This virtual hosts overrules the default Apache `ErrorDocument` directive.

```
<VirtualHost 83.217.76.245:80>
ServerName cobbaut.be
ServerAlias www.cobbaut.be
DocumentRoot /home/paul/public_html
ErrorLog /home/paul/logs/error_log
CustomLog /home/paul/logs/access_log common
ScriptAlias /cgi-bin/ /home/paul/cgi-bin/
<Directory /home/paul/public_html>
    Options Indexes IncludesNOEXEC FollowSymLinks
    allow from all
</Directory>
ErrorDocument 404 http://www.cobbaut.be/cobbaut.php
</VirtualHost>
```

20.10. aliases and redirects

Apache supports aliases for directories, like this example shows.

```
Alias /paul/ "/home/paul/public_html/"
```

Similarly, content can be redirected to another website or web server.

```
Redirect permanent /foo http://www.foo.com/bar
```

20.11. more on .htaccess

You can do much more with `.htaccess`. One example is to use .htaccess to prevent people from certain domains to access your website. Like in this case, where a number of referer spammers are blocked from the website.

```
student@linux:~/cobbaut.be$ cat .htaccess
# Options +FollowSymlinks
RewriteEngine On
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-adipex.fw.nu.*$ [OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-levitra.asso.ws.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-tramadol.fw.nu.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-viagra.lookin.at.*$ [NC,OR]
...
RewriteCond %{HTTP_REFERER} ^http://(www\.)?www.healthinsurancehelp.net.*$ [NC]
RewriteRule .* - [F,L]
student@linux:~/cobbaut.be$
```

20.12. traffic

Apache keeps a log of all visitors. The `webalizer` is often used to parse this log into nice html statistics.

20.13. self signed cert on Debian

Below is a very quick guide on setting up Apache2 on Debian 7 with a self-signed certificate.

Chances are these packages are already installed.

```
root@linux:~# aptitude install apache2 openssl
No packages will be installed, upgraded, or removed.
0 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B of archives. After unpacking 0 B will be used.
```

Create a directory to store the certs, and use `openssl` to create a self signed cert that is valid for 999 days.

```
root@linux:~# mkdir /etc/ssl/localcerts
root@linux:~# openssl req -new -x509 -days 999 -nodes -out /etc/ssl/local\
certs/apache.pem -keyout /etc/ssl/localcerts/apache.key
Generating a 2048 bit RSA private key
...
...
writing new private key to '/etc/ssl/localcerts/apache.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BE
State or Province Name (full name) [Some-State]:Antwerp
Locality Name (eg, city) []:Antwerp
Organization Name (eg, company) [Internet Widgits Pty Ltd]:linux-training.be
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Paul
Email Address []:
```

A little security never hurt anyone.

```
root@linux:~# ls -l /etc/ssl/localcerts/
total 8
-rw-r--r-- 1 root root 1704 Sep 16 18:24 apache.key
-rw-r--r-- 1 root root 1302 Sep 16 18:24 apache.pem
root@linux:~# chmod 600 /etc/ssl/localcerts/*
root@linux:~# ls -l /etc/ssl/localcerts/
total 8
-rw------- 1 root root 1704 Sep 16 18:24 apache.key
-rw------- 1 root root 1302 Sep 16 18:24 apache.pem
```

Enable the `apache ssl mod`.

```
root@linux:~# 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.
To activate the new configuration, you need to run:
  service apache2 restart
```

Create the website configuration.

```
root@linux:~# vi /etc/apache2/sites-available/choochoos
root@linux:~# cat /etc/apache2/sites-available/choochoos
<VirtualHost *:7000>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/choochoos
        SSLEngine On
        SSLCertificateFile /etc/ssl/localcerts/apache.pem
        SSLCertificateKeyFile /etc/ssl/localcerts/apache.key
</VirtualHost>
root@linux:~#
```

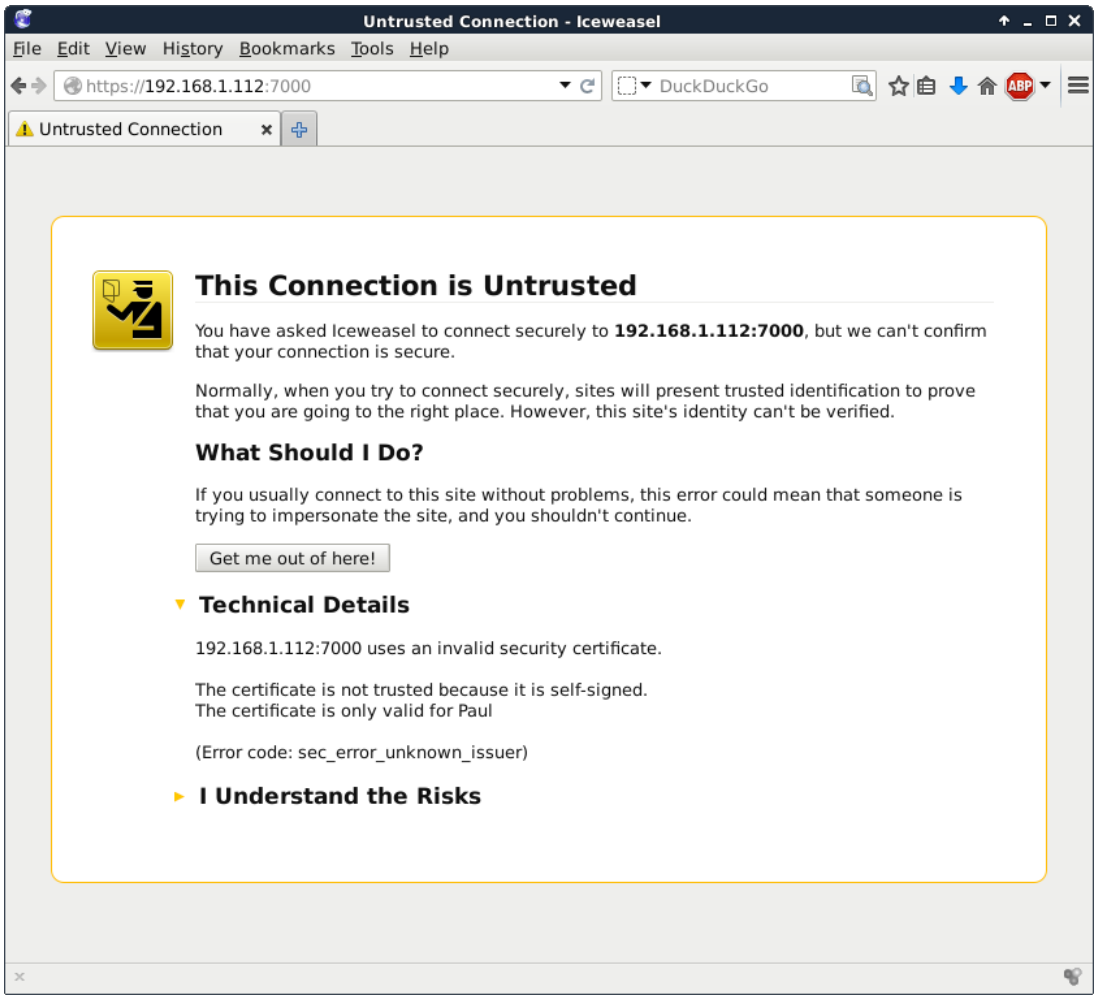
And create the website itself.

```
root@linux:/var/www/choochoos# vi index.html
root@linux:/var/www/choochoos# cat index.html
Choo Choo HTTPS secured model train Choo Choo
```

Enable the website and restart (or reload) apache2.

```
root@linux:/var/www/choochoos# a2ensite choochoos
Enabling site choochoos.
To activate the new configuration, you need to run:
  service apache2 reload
root@linux:/var/www/choochoos# service apache2 restart
Restarting web server: apache2 ... waiting .
```

Chances are your browser will warn you about the self signed certificate.



20.14. self signed cert on RHEL/CentOS

Below is a quick way to create a self signed cert for https on RHEL/CentOS. You may need these packages:

```
[root@paulserver ~]# yum install httpd openssl mod_ssl
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.belnet.be
 * extras: ftp.belnet.be
 * updates: mirrors.vooservers.com
base                                                     | 3.7 kB     00:00
Setting up Install Process
Package httpd-2.2.15-31.el6.centos.x86_64 already installed and latest version
Package openssl-1.0.1e-16.el6_5.15.x86_64 already installed and latest version
Package 1:mod_ssl-2.2.15-31.el6.centos.x86_64 already ins ... and latest version
Nothing to do
```

We use `openssl` to create the certificate.

```
[root@paulserver ~]# mkdir certs
[root@paulserver ~]# cd certs
[root@paulserver certs]# openssl genrsa -out ca.key 2048
```

```
Generating RSA private key, 2048 bit long modulus
.........+++
.....................................................+++
e is 65537 (0×10001)
[root@paulserver certs]# openssl req -new -key ca.key -out ca.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:BE
State or Province Name (full name) []:antwerp
Locality Name (eg, city) [Default City]:antwerp
Organization Name (eg, company) [Default Company Ltd]:antwerp
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:paulserver
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@paulserver certs]# openssl x509 -req -days 365 -in ca.csr -signkey ca.ke\
y -out ca.crt
Signature ok
subject=/C=BE/ST=antwerp/L=antwerp/O=antwerp/CN=paulserver
Getting Private key
```

We copy the keys to the right location (You may be missing SELinux info here).

```
[root@paulserver certs]# cp ca.crt /etc/pki/tls/certs/
[root@paulserver certs]# cp ca.key ca.csr /etc/pki/tls/private/
```

We add the location of our keys to this file, and also add the `NameVirtualHost *:443` directive.

```
[root@paulserver certs]# vi /etc/httpd/conf.d/ssl.conf
[root@paulserver certs]# grep ^SSLCerti /etc/httpd/conf.d/ssl.conf
SSLCertificateFile /etc/pki/tls/certs/ca.crt
SSLCertificateKeyFile /etc/pki/tls/private/ca.key
```

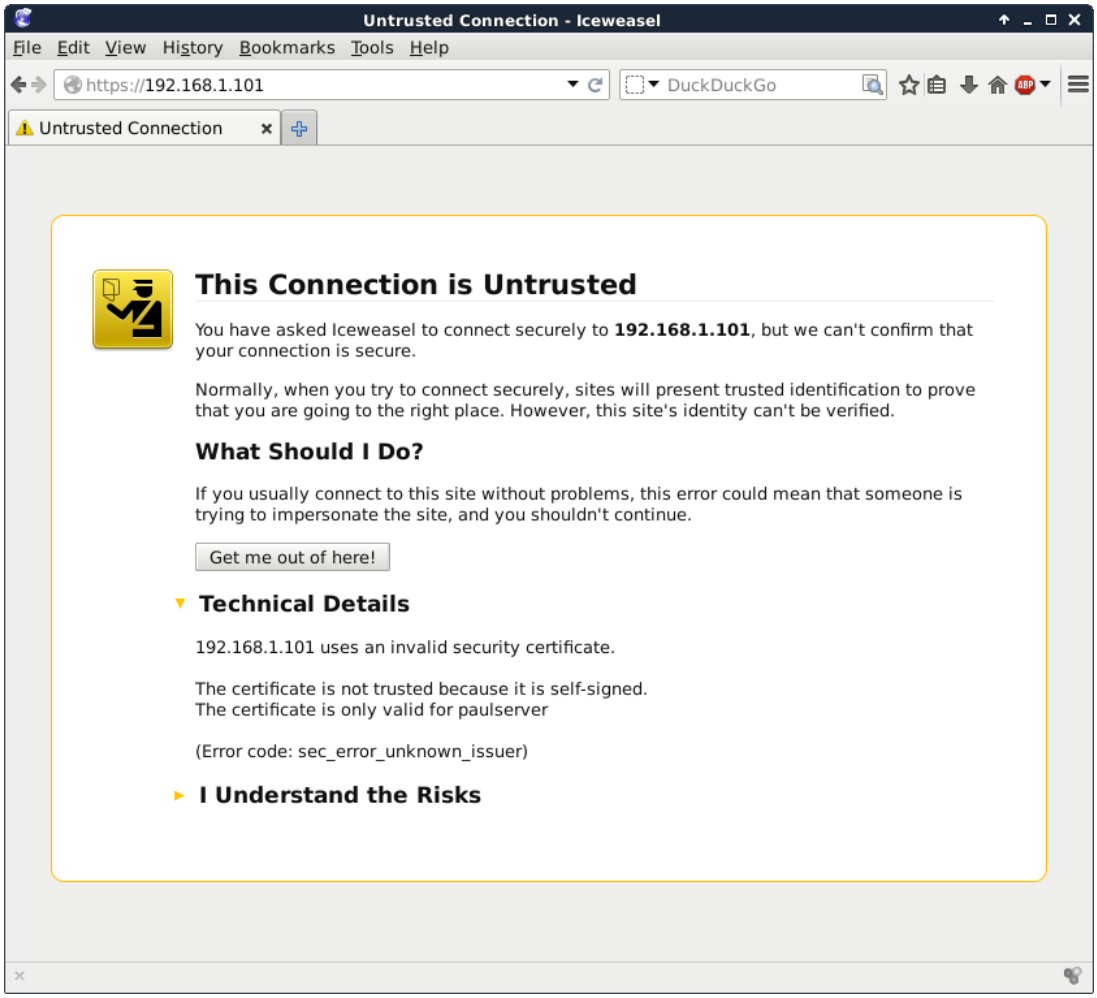
Create a website configuration.

```
[root@paulserver certs]# vi /etc/httpd/conf.d/choochoos.conf
[root@paulserver certs]# cat /etc/httpd/conf.d/choochoos.conf
<VirtualHost *:443>
        SSLEngine on
        SSLCertificateFile /etc/pki/tls/certs/ca.crt
        SSLCertificateKeyFile /etc/pki/tls/private/ca.key
        DocumentRoot /var/www/choochoos
        ServerName paulserver
</VirtualHost>
[root@paulserver certs]#
```

Create a simple website and restart apache.

```
[root@paulserver certs]# mkdir /var/www/choochoos
[root@paulserver certs]# echo HTTPS model train choochoos > /var/www/choochoos/\
index.html
[root@paulserver httpd]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

And your browser will probably warn you that this certificate is self signed.



20.15. practice: apache

1. Verify that Apache is installed and running.
2. Browse to the Apache HTML manual.
3. Create three virtual hosts that listen on ports 8472, 31337 and 1201. Test that it all works.
4. Create three named virtual hosts startrek.local, starwars.local and stargate.local. Test that it all works.
5. Create a virtual hosts that listens on another ip-address.
6. Protect one of your websites with a user/password combo.

21. scripting loops

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

21.1. test [ ]

The `test` command can test whether something is true or false. Let's start by testing whether 10 is greater than 55.

```
[student@linux ~]$ test 10 -gt 55 ; echo $?
1
[student@linux ~]$
```

The test command returns 1 if the test fails. And as you see in the next screenshot, test returns 0 when a test succeeds.

```
[student@linux ~]$ test 56 -gt 55 ; echo $?
0
[student@linux ~]$
```

If you prefer true and false, then write the test like this.

```
[student@linux ~]$ test 56 -gt 55 && echo true || echo false
true
[student@linux ~]$ test 6 -gt 55 && echo true || echo false
false
```

The test command can also be written as square brackets, the screenshot below is identical to the one above.

```
[student@linux ~]$ [ 56 -gt 55 ] && echo true || echo false
true
[student@linux ~]$ [ 6 -gt 55 ] && echo true || echo false
false
```

Below are some example tests. Take a look at `man test` to see more options for tests.

```
[ -d foo ]              Does the directory foo exist ?
[ -e bar ]              Does the file bar exist ?
[ '/etc' = $PWD ]       Is the string /etc equal to the variable $PWD ?
[ $1 =≠ 'secret' ]      Is the first parameter different from secret ?
[ 55 -lt $bar ]         Is 55 less than the value of $bar ?
[ $foo -ge 1000 ]       Is the value of $foo greater or equal to 1000 ?
[ "abc" < $bar ]        Does abc sort before the value of $bar ?
[ -f foo ]              Is foo a regular file ?
[ -r bar ]              Is bar a readable file ?
[ foo -nt bar ]         Is file foo newer than file bar ?
[ -o nounset ]          Is the shell option nounset set ?
```

Tests can be combined with logical AND and OR.

```
student@linux:~$ [ 66 -gt 55 -a 66 -lt 500 ] && echo true || echo false
true
student@linux:~$ [ 66 -gt 55 -a 660 -lt 500 ] && echo true || echo false
false
student@linux:~$ [ 66 -gt 55 -o 660 -lt 500 ] && echo true || echo false
true
```

21.2. if then else

The `if then else` construction is about choice. If a certain condition is met, then execute something, else execute something else. The example below tests whether a file exists, and if the file exists then a proper message is echoed.

```
#!/bin/bash

if [ -f isit.txt ]
then echo isit.txt exists!
else echo isit.txt not found!
fi
```

If we name the above script 'choice', then it executes like this.

```
[student@linux scripts]$ ./choice
isit.txt not found!
[student@linux scripts]$ touch isit.txt
[student@linux scripts]$ ./choice
isit.txt exists!
[student@linux scripts]$
```

21.3. if then elif

You can nest a new `if` inside an `else` with `elif`. This is a simple example.

```
#!/bin/bash
count=42
if [ $count -eq 42 ]
then
  echo "42 is correct."
elif [ $count -gt 42 ]
then
  echo "Too much."
else
  echo "Not enough."
fi
```

21.4. for loop

The example below shows the syntax of a classical `for loop` in bash.

```
for i in 1 2 4
do
   echo $i
done
```

An example of a `for loop` combined with an embedded shell.

```
#!/bin/ksh
for counter in `seq 1 20`
do
   echo counting from 1 to 20, now at $counter
   sleep 1
done
```

The same example as above can be written without the embedded shell using the bash `{from..to}` shorthand.

```
#!/bin/bash
for counter in {1..20}
do
   echo counting from 1 to 20, now at $counter
   sleep 1
done
```

This `for loop` uses file globbing (from the shell expansion). Putting the instruction on the command line has identical functionality.

```
kahlan@solexp11$ ls
count.ksh  go.ksh
kahlan@solexp11$ for file in *.ksh ; do cp $file $file.backup ; done
kahlan@solexp11$ ls
count.ksh  count.ksh.backup  go.ksh  go.ksh.backup
```

21.5. while loop

Below a simple example of a `while loop`.

```
i=100;
while [ $i -ge 0 ] ;
do
   echo Counting down, from 100 to 0, now at $i;
   let i--;
done
```

Endless loops can be made with `while true` or `while :` , where the `colon` is the equivalent of `no operation` in the `Korn` and `bash` shells.

```
#!/bin/ksh
# endless loop
while :
do
 echo hello
 sleep 1
done
```

21.6. until loop

Below a simple example of an `until loop`.

```
let i=100;
until [ $i -le 0 ] ;
do
   echo Counting down, from 100 to 1, now at $i;
   let i--;
done
```

21.7. practice: scripting tests and loops

1. Write a script that uses a `for` loop to count from 3 to 7.
2. Write a script that uses a `for` loop to count from 1 to 17000.
3. Write a script that uses a `while` loop to count from 3 to 7.
4. Write a script that uses an `until` loop to count down from 8 to 4.
5. Write a script that counts the number of files ending in `.txt` in the current directory.
6. Wrap an `if` statement around the script so it is also correct when there are zero files ending in `.txt`.

21.8. solution: scripting tests and loops

1. Write a script that uses a `for` loop to count from 3 to 7.

```
#!/bin/bash

for i in 3 4 5 6 7
do
   echo "Counting from 3 to 7, now at ${i}"
done
```

2. Write a script that uses a `for` loop to count from 1 to 17000.

```
#!/bin/bash

for i in `seq 1 17000`
do
    echo "Counting from 1 to 17000, now at ${i}"
done
```

3. Write a script that uses a `while` loop to count from 3 to 7.

```
#!/bin/bash

i=3
while [ $i -le 7 ]
do
 echo "Counting from 3 to 7, now at ${i}"
 let i=i+1
done
```

4. Write a script that uses an `until` loop to count down from 8 to 4.

```
#!/bin/bash

i=8
until [ $i -lt 4 ]
do
 echo "Counting down from 8 to 4, now at ${i}"
 let i=i-1
done
```

5. Write a script that counts the number of files ending in `.txt` in the current directory.

```
#!/bin/bash

let i=0
for file in *.txt
do
    let i++
done
echo "There are ${i} files ending in .txt"
```

6. Wrap an `if` statement around the script so it is also correct when there are zero files ending in `.txt`.

```
#!/bin/bash

ls *.txt > /dev/null 2>&1
if [ $? -ne 0 ]
then echo "There are 0 files ending in .txt"
else
    let i=0
    for file in *.txt
    do
        let i++
    done
    echo "There are ${i} files ending in .txt"
fi
```

22. scripting parameters

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

22.1. script parameters

A `bash` shell script can have parameters. The numbering you see in the script below continues if you have more parameters. You also have special parameters containing the number of parameters, a string of all of them, and also the process id, and the last return code. The man page of `bash` has a full list.

```
#!/bin/bash
echo The first argument is $1
echo The second argument is $2
echo The third argument is $3

echo \$ $$  PID of the script
echo \# $#  count arguments
echo \? $?  last return code
echo \* $*  all the arguments
```

Below is the output of the script above in action.

```
[student@linux scripts]$ ./pars one two three
The first argument is one
The second argument is two
The third argument is three
$ 5610 PID of the script
# 3 count arguments
? 0 last return code
* one two three all the arguments
```

Once more the same script, but with only two parameters.

```
[student@linux scripts]$ ./pars 1 2
The first argument is 1
The second argument is 2
The third argument is
$ 5612 PID of the script
# 2 count arguments
? 0 last return code
* 1 2 all the arguments
[student@linux scripts]$
```

Here is another example, where we use `$0`. The `$0` parameter contains the name of the script.

```
student@linux~$ cat myname
echo this script is called $0
student@linux~$ ./myname
this script is called ./myname
student@linux~$ mv myname test42
student@linux~$ ./test42
this script is called ./test42
```

22.2. shift through parameters

The `shift` statement can parse all `parameters` one by one. This is a sample script.

```
kahlan@solexp11$ cat shift.ksh
#!/bin/ksh

if [ "$#" == "0" ]
 then
  echo You have to give at least one parameter.
  exit 1
fi

while (( $# ))
 do
  echo You gave me $1
  shift
 done
```

Below is some sample output of the script above.

```
kahlan@solexp11$ ./shift.ksh one
You gave me one
kahlan@solexp11$ ./shift.ksh one two three 1201 "33 42"
You gave me one
You gave me two
You gave me three
You gave me 1201
You gave me 33 42
kahlan@solexp11$ ./shift.ksh
You have to give at least one parameter.
```

22.3. runtime input

You can ask the user for input with the `read` command in a script.

```
#!/bin/bash
echo -n Enter a number:
read number
```

22.4. sourcing a config file

The `source` (as seen in the shell chapters) can be used to source a configuration file.

Below a sample configuration file for an application.

```
[student@linux scripts]$ cat myApp.conf
# The config file of myApp

# Enter the path here
myAppPath=/var/myApp

# Enter the number of quines here
quines=5
```

And here an application that uses this file.

```
[student@linux scripts]$ cat myApp.bash
#!/bin/bash
#
# Welcome to the myApp application
#

. ./myApp.conf

echo There are $quines quines
```

The running application can use the values inside the sourced configuration file.

```
[student@linux scripts]$ ./myApp.bash
There are 5 quines
[student@linux scripts]$
```

22.5. get script options with getopts

The `getopts` function allows you to parse options given to a command. The following script allows for any combination of the options a, f and z.

```
kahlan@solexp11$ cat options.ksh
#!/bin/ksh

while getopts ":afz" option;
do
 case $option in
  a)
   echo received -a
   ;;
  f)
   echo received -f
   ;;
  z)
   echo received -z
   ;;
  *)
   echo "invalid option -$OPTARG"
```

```
   ;;
 esac
done
```

This is sample output from the script above. First we use correct options, then we enter twice an invalid option.

```
kahlan@solexp11$ ./options.ksh
kahlan@solexp11$ ./options.ksh -af
received -a
received -f
kahlan@solexp11$ ./options.ksh -zfg
received -z
received -f
invalid option -g
kahlan@solexp11$ ./options.ksh -a -b -z
received -a
invalid option -b
received -z
```

You can also check for options that need an argument, as this example shows.

```
kahlan@solexp11$ cat argoptions.ksh
#!/bin/ksh

while getopts ":af:z" option;
do
 case $option in
  a)
   echo received -a
   ;;
  f)
   echo received -f with $OPTARG
   ;;
  z)
   echo received -z
   ;;
  :)
   echo "option -$OPTARG needs an argument"
   ;;
  *)
   echo "invalid option -$OPTARG"
   ;;
 esac
done
```

This is sample output from the script above.

```
kahlan@solexp11$ ./argoptions.ksh -a -f hello -z
received -a
received -f with hello
received -z
kahlan@solexp11$ ./argoptions.ksh -zaf 42
received -z
received -a
received -f with 42
kahlan@solexp11$ ./argoptions.ksh -zf
received -z
option -f needs an argument
```

22.6. get shell options with shopt

You can toggle the values of variables controlling optional shell behaviour with the `shopt` built-in shell command. The example below first verifies whether the cdspell option is set; it is not. The next shopt command sets the value, and the third shopt command verifies that the option really is set. You can now use minor spelling mistakes in the cd command. The man page of bash has a complete list of options.

```
student@linux:~$ shopt -q cdspell ; echo $?
1
student@linux:~$ shopt -s cdspell
student@linux:~$ shopt -q cdspell ; echo $?
0
student@linux:~$ cd /Etc
/etc
```

22.7. practice: parameters and options

1. Write a script that receives four parameters, and outputs them in reverse order.

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

3. Write a script that asks for a filename. Verify existence of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

4. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

22.8. solution: parameters and options

1. Write a script that receives four parameters, and outputs them in reverse order.

```
echo $4 $3 $2 $1
```

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

```
#!/bin/bash

if [ -f $1 ]
then echo $1 exists!
else echo $1 not found!
fi

if [ -f $2 ]
then echo $2 exists!
else echo $2 not found!
fi
```

3. Write a script that asks for a filename. Verify existence of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

4. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

Part VI.

Webserver hardening

23. introduction to SELinux

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

Security Enhanced Linux or `SELinux` is a set of modifications developed by the United States National Security Agency (NSA) to provide a variety of security policies for Linux. SELinux was released as open source at the end of 2000. Since kernel version 2.6 it is an integrated part of Linux.

SELinux offers security! SELinux can control what kind of access users have to files and processes. Even when a file received `chmod 777`, SELinux can still prevent applications from accessing it (Unix file permissions are checked first!). SELinux does this by placing users in `roles` that represent a security context. Administrators have very strict control on access permissions granted to roles.

SELinux is present in the latest versions of Red Hat Enterprise Linux, Debian, CentOS, Fedora, and many other distributions..

23.1. selinux modes

`selinux` knows three modes: enforcing, permissive and disabled. The `enforcing` mode will enforce policies, and may deny access based on `selinux rules`. The `permissive` mode will not enforce policies, but can still log actions that would have been denied in `enforcing` mode. The `disabled` mode disables `selinux`.

23.2. logging

Verify that `syslog` is running and activated on boot to enable logging of deny messages in `/var/log/messages`.

```
[root@linux ~]# chkconfig --list syslog
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Verify that `auditd` is running and activated on boot to enable logging of easier to read messages in `/var/log/audit/audit.log`.

```
[root@linux ~]# chkconfig --list auditd
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

If not activated, then run `chkconfig --levels 2345 auditd on` and `service auditd start`.

```
[root@linux ~]# service auditd status
auditd (pid  1660) is running...
[root@linux ~]# service syslog status
syslogd (pid  1688) is running...
klogd (pid  1691) is running...
```

The /var/log/messages log file will tell you that selinux is disabled.

```
root@linux:~# grep -i selinux /var/log/messages
Jun 25 15:59:34 deb106 kernel: [    0.084083] SELinux:  Disabled at boot.
```

Or that it is enabled.

```
root@linux:~# grep SELinux /var/log/messages | grep -i Init
Jun 25 15:09:52 deb106 kernel: [    0.084094] SELinux:  Initializing.
```

23.3. activating selinux

On RHEL you can use the GUI tool to activate selinux, on Debian there is the selinux-activate command. Activation requires a reboot.

```
root@linux:~# selinux-activate
Activating SE Linux
Searching for GRUB installation directory ... found: /boot/grub
Searching for default file ... found: /boot/grub/default
Testing for an existing GRUB menu.lst file ... found: /boot/grub/menu.lst
Searching for splash image ... none found, skipping ...
Found kernel: /boot/vmlinuz-2.6.26-2-686
Updating /boot/grub/menu.lst ... done

SE Linux is activated.  You may need to reboot now.
```

23.4. getenforce

Use getenforce to verify whether selinux is enforced, disabled or permissive.

```
[root@linux ~]# getenforce
Permissive
```

The /selinux/enforce file contains 1 when enforcing, and 0 when permissive mode is active.

```
root@fedora13 ~# cat /selinux/enforce
1root@fedora13 ~#
```

23.5. setenforce

You can use setenforce to switch between the Permissive or the Enforcing state once selinux is activated..

```
[root@linux ~]# setenforce Enforcing
[root@linux ~]# getenforce
Enforcing
[root@linux ~]# setenforce Permissive
[root@linux ~]# getenforce
Permissive
```

Or you could just use 0 and 1 as argument.

```
[root@linux ~]# setenforce 1
[root@linux ~]# getenforce
Enforcing
[root@linux ~]# setenforce 0
[root@linux ~]# getenforce
Permissive
[root@linux ~]#
```

23.6. sestatus

You can see the current `selinux` status and policy with the `sestatus` command.

```
[root@linux ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   permissive
Mode from config file:          permissive
Policy version:                 21
Policy from config file:        targeted
```

23.7. policy

Most Red Hat server will have the `targeted` policy. Only NSA/FBI/CIA/DOD/HLS use the `mls` policy.

The targted policy will protect hundreds of processes, but lets other processes run 'unconfined' (= they can do anything).

23.8. /etc/selinux/config

The main configuration file for `selinux` is `/etc/selinux/config`. When in `permissive` mode, the file looks like this.

The targeted policy is selected in `/etc/selinux/config`.

```
[root@linux ~]# cat /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

23.9. DAC or MAC

Standard Unix permissions use `Discretionary Access Control` to set permissions on files. This means that a user that owns a file, can make it world readable by typing `chmod 777 $file`.

With `selinux` the kernel will enforce `Mandatory Access Control` which strictly controls what processes or threads can do with files (superseding DAC). Processes are confined by the kernel to the minimum access they require.

SELinux MAC is about labeling and type enforcing! Files, processes, etc are all labeled with an SELinux context. For files, these are extended attributes, for processes this is managed by the kernel.

The format of the labels is as follows:

```
user:role:type:(level)
```

We only use the `type` label in the targeted policy.

23.10. ls -Z

To see the DAC permissions on a file, use `ls -l` to display user and group `owner` and permissions.

For MAC permissions there is new `-Z` option added to `ls`. The output shows that file in `/root` have a XXXtype of `admin_home_t`.

```
[root@linux ~]# ls -Z
-rw-------. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log.syslog

[root@linux ~]# useradd -m -s /bin/bash pol
[root@linux ~]# ls -Z /home/pol/.bashrc
-rw-r--r--. pol pol unconfined_u:object_r:user_home_t:s0 /home/pol/.bashrc
```

23.11. -Z

There are also some other tools with the -Z switch:

```
mkdir -Z
cp -Z
ps -Z
netstat -Z
...
```

23.12. /selinux

When selinux is active, there is a new virtual file system named `/selinux`. (You can compare it to /proc and /dev.)

```
[root@linux ~]# ls -l /selinux/
total 0
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 access
dr-xr-xr-x.  2 root root    0 Apr 12 19:40 avc
dr-xr-xr-x.  2 root root    0 Apr 12 19:40 booleans
-rw-r--r--.  1 root root    0 Apr 12 19:40 checkreqprot
dr-xr-xr-x. 83 root root    0 Apr 12 19:40 class
--w-------.  1 root root    0 Apr 12 19:40 commit_pending_bools
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 context
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 create
-r--r--r--.  1 root root    0 Apr 12 19:40 deny_unknown
--w-------.  1 root root    0 Apr 12 19:40 disable
-rw-r--r--.  1 root root    0 Apr 12 19:40 enforce
dr-xr-xr-x.  2 root root    0 Apr 12 19:40 initial_contexts
-rw-------.  1 root root    0 Apr 12 19:40 load
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 member
-r--r--r--.  1 root root    0 Apr 12 19:40 mls
crw-rw-rw-.  1 root root 1, 3 Apr 12 19:40 null
-r--------.  1 root root    0 Apr 12 19:40 policy
dr-xr-xr-x.  2 root root    0 Apr 12 19:40 policy_capabilities
-r--r--r--.  1 root root    0 Apr 12 19:40 policyvers
-r--r--r--.  1 root root    0 Apr 12 19:40 reject_unknown
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 relabel
-r--r--r--.  1 root root    0 Apr 12 19:40 status
-rw-rw-rw-.  1 root root    0 Apr 12 19:40 user
```

Although some files in `/selinux` appear wih size 0, they often contain a boolean value. Check `/selinux/enforce` to see if selinux is running in enforced mode.

```
[root@linux ~]# ls -l /selinux/enforce
-rw-r--r-- 1 root root 0 Apr 29 08:21 /selinux/enforce
[root@linux ~]# echo $(cat /selinux/enforce)
1
```

23.13. identity

The `SELinux Identity` of a user is distinct from the user ID. An identity is part of a security context, and (via domains) determines what you can do. The screenshot shows user `root` having identity `user_u`.

```
[root@linux ~]# id -Z
user_u:system_r:unconfined_t
```

23.14. role

The `selinux role` defines the domains that can be used. A `role` is denied to enter a domain, unless the `role` is explicitely authorized to do so.

23.15. type (or domain)

The `selinux context` is the security context of a process. An `selinux type` determines what a process can do. The screenshot shows init running in type `init_t` and the mingetty's running in type `getty_t`.

```
[root@linux ~]# ps fax -Z | grep /sbin/init
system_u:system_r:init_t:s0         1 ?        Ss     0:00 /sbin/init
[root@linux ~]# ps fax -Z | grep getty_t
system_u:system_r:getty_t:s0  1307 tty1   Ss+   0:00 /sbin/mingetty /dev/tty1
system_u:system_r:getty_t:s0  1309 tty2   Ss+   0:00 /sbin/mingetty /dev/tty2
system_u:system_r:getty_t:s0  1311 tty3   Ss+   0:00 /sbin/mingetty /dev/tty3
system_u:system_r:getty_t:s0  1313 tty4   Ss+   0:00 /sbin/mingetty /dev/tty4
system_u:system_r:getty_t:s0  1320 tty5   Ss+   0:00 /sbin/mingetty /dev/tty5
system_u:system_r:getty_t:s0  1322 tty6   Ss+   0:00 /sbin/mingetty /dev/tty6
```

The `selinux type` is similar to an `selinux domain`, but refers to directories and files instead of processes.

Hundreds of binaries also have a type:

```
[root@linux sbin]# ls -lZ useradd usermod userdel httpd postcat postfix
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t:s0 httpd
-rwxr-xr-x. root root system_u:object_r:postfix_master_exec_t:s0 postcat
-rwxr-xr-x. root root system_u:object_r:postfix_master_exec_t:s0 postfix
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 useradd
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 userdel
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 usermod
```

Ports also have a context.

```
[root@linux sbin]# netstat -nptlZ | tr -s ' ' | cut -d' ' -f6-

Foreign Address State PID/Program name Security Context
LISTEN 1096/rpcbind system_u:system_r:rpcbind_t:s0
LISTEN 1208/sshd system_u:system_r:sshd_t:s0-s0:c0.c1023
LISTEN 1284/master system_u:system_r:postfix_master_t:s0
LISTEN 1114/rpc.statd system_u:system_r:rpcd_t:s0
LISTEN 1096/rpcbind system_u:system_r:rpcbind_t:s0
LISTEN 1666/httpd unconfined_u:system_r:httpd_t:s0
LISTEN 1208/sshd system_u:system_r:sshd_t:s0-s0:c0.c1023
LISTEN 1114/rpc.statd system_u:system_r:rpcd_t:s0
LISTEN 1284/master system_u:system_r:postfix_master_t:s0
```

You can also get a list of ports that are managed by SELinux:

```
[root@linux ~]# semanage port -l | tail
xfs_port_t                     tcp      7100
xserver_port_t                 tcp      6000-6150
zabbix_agent_port_t            tcp      10050
zabbix_port_t                  tcp      10051
zarafa_port_t                  tcp      236, 237
zebra_port_t                   tcp      2600-2604, 2606
zebra_port_t                   udp      2600-2604, 2606
zented_port_t                  tcp      1229
zented_port_t                  udp      1229
zope_port_t                    tcp      8021
```

23.16. security context

The combination of identity, role and domain or type make up the `selinux security context`. The `id` will show you your security context in the form identity:role:domain.

```
[student@linux ~]$ id | cut -d' ' -f4
context=user_u:system_r:unconfined_t
```

The `ls -Z` command shows the security context for a file in the form identity:role:type.

```
[student@linux ~]$ ls -Z test
-rw-rw-r--  paul paul user_u:object_r:user_home_t      test
```

The security context for processes visible in /proc defines both the type (of the file in /proc) and the domain (of the running process). Let's take a look at the init process and /proc/1/ .

The init process runs in domain `init_t`.

```
[root@linux ~]# ps -ZC init
LABEL                             PID TTY          TIME CMD
system_u:system_r:init_t            1 ?        00:00:01 init
```

The `/proc/1/` directory, which identifies the `init` process, has type `init_t`.

```
[root@linux ~]# ls -Zd /proc/1/
dr-xr-xr-x  root root system_u:system_r:init_t         /proc/1/
```

It is not a coincidence that the domain of the `init` process and the type of `/proc/1/` are both `init_t`.

Don't try to use `chcon` on /proc! It will not work.

23.17. transition

An `selinux transition` (aka an selinux labelling) determines the security context that will be assigned. A transition of process domains is used when you execute a process. A transition of file type happens when you create a file.

An example of file type transition.

```
[pol@linux ~]$ touch test /tmp/test
[pol@linux ~]$ ls -Z test
-rw-rw-r--. pol pol unconfined_u:object_r:user_home_t:s0 test
[pol@linux ~]$ ls -Z /tmp/test
-rw-rw-r--. pol pol unconfined_u:object_r:user_tmp_t:s0 /tmp/test
```

23.18. extended attributes

Extended attributes are used by `selinux` to store security contexts. These attributes can be viewed with `ls` when `selinux` is running.

```
[root@linux home]# ls --context
drwx------  paul paul system_u:object_r:user_home_dir_t paul
drwxr-xr-x  root root user_u:object_r:user_home_dir_t  project42
drwxr-xr-x  root root user_u:object_r:user_home_dir_t  project55
[root@linux home]# ls -Z
drwx------  paul paul system_u:object_r:user_home_dir_t paul
drwxr-xr-x  root root user_u:object_r:user_home_dir_t  project42
drwxr-xr-x  root root user_u:object_r:user_home_dir_t  project55
[root@linux home]#
```

When selinux is not running, then `getfattr` is the tool to use.

```
[root@linux etc]# getfattr -m . -d hosts
# file: hosts
security.selinux="system_u:object_r:etc_t:s0\000"
```

23.19. process security context

A new option is added to `ps` to see the selinux security context of processes.

```
[root@linux etc]# ps -ZC mingetty
LABEL                             PID TTY          TIME CMD
system_u:system_r:getty_t        2941 tty1     00:00:00 mingetty
system_u:system_r:getty_t        2942 tty2     00:00:00 mingetty
```

23.20. chcon

Use `chcon` to change the selinux security context.

This example shows how to use `chcon` to change the `type` of a file.

```
[root@linux ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--  root root user_u:object_r:httpd_sys_content_t /var/www/html/test4\
2.txt
[root@linux ~]# chcon -t samba_share_t /var/www/html/test42.txt
[root@linux ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--  root root user_u:object_r:samba_share_t   /var/www/html/test42.txt
```

Be sure to read `man chcon`.

23.21. an example

The Apache2 webserver is by default targeted with SELinux. The next screenshot shows that any file created in /var/www/html will by default get the httpd_sys_content_t type.

```
[root@linux ~]# touch /var/www/html/test42.txt
[root@linux ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/h\
tml/test42.txt
```

Files created elsewhere do not get this type.

```
[root@linux ~]# touch /root/test42.txt
[root@linux ~]# ls -Z /root/test42.txt
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 /root/test42.txt
```

Make sure Apache2 runs.

```
[root@linux ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

Will this work ? Yes it does.

```
[root@linux ~]# wget http://localhost/test42.txt
--2014-04-12 20:56:47--  http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: “test42.txt”
...
```

Why does this work ? Because Apache2 runs in the httpd_t domain and the files in /var/www/html have the httpd_sys_content_t type.

```
[root@linux ~]# ps -ZC httpd | head -4
LABEL                              PID TTY          TIME CMD
unconfined_u:system_r:httpd_t:s0 1666 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 1668 ?        00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 1669 ?        00:00:00 httpd
```

So let’s set SELinux to enforcing and change the type of this file.

```
[root@linux ~]# chcon -t samba_share_t /var/www/html/test42.txt
[root@linux ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--. root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/t\
est42.txt
[root@linux ~]# setenforce 1
[root@linux ~]# getenforce
Enforcing
```

There are two possibilities now: either it works, or it fails. It works when selinux is in permissive mode, it fails when in enforcing mode.

```
[root@linux ~]# wget http://localhost/test42.txt
--2014-04-12 21:05:02--  http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-04-12 21:05:02 ERROR 403: Forbidden.
```

The log file gives you a cryptic message...

```
[root@linux ~]# tail -3 /var/log/audit/audit.log
type=SYSCALL msg=audit(1398200702.803:64): arch=c000003e syscall=4 succ\
ess=no exit=-13 a0=7f5fbc334d70 a1=7fff553b4f10 a2=7fff553b4f10 a3=0 it\
ems=0 ppid=1666 pid=1673 auid=500 uid=48 gid=48 euid=48 suid=48 fsuid=4\
8 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd" exe="/usr/sbin\
/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1398200702.804:65): avc:  denied  { getattr } for  p\
id=1673 comm="httpd" path="/var/www/html/test42.txt" dev=dm-0 ino=26324\
1 scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:objec\
t_r:samba_share_t:s0 tclass=file
type=SYSCALL msg=audit(1398200702.804:65): arch=c000003e syscall=6 succ\
ess=no exit=-13 a0=7f5fbc334e40 a1=7fff553b4f10 a2=7fff553b4f10 a3=1 it\
ems=0 ppid=1666 pid=1673 auid=500 uid=48 gid=48 euid=48 suid=48 fsuid=4\
8 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd" exe="/usr/sbin\
/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

And `/var/log/messages` mentions nothing of the failed download.

23.22. setroubleshoot

The log file above was not very helpful, but these two packages can make your life much easier.

```
[root@linux ~]# yum -y install setroubleshoot setroubleshoot-server
```

You need to `reboot` for this to work...

So we reboot, restart the httpd server, reactive SELinux Enforce, and do the wget again... and it fails (because of SELinux).

```
[root@linux ~]# service httpd restart
Stopping httpd:                                            [FAILED]
Starting httpd:                                            [  OK  ]
[root@linux ~]# getenforce
Permissive
[root@linux ~]# setenforce  1
[root@linux ~]# getenforce
Enforcing
[root@linux ~]# wget http://localhost/test42.txt
--2014-04-12 21:44:13--  http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-04-12 21:44:13 ERROR 403: Forbidden.
```

The `/var/log/audit/` is still not out best friend, but take a look at `/var/log/messages`.

```
[root@linux ~]# tail -2 /var/log/messages
Apr 12 21:44:16  centos65  setroubleshoot: SELinux is preventing /usr/sbin/h\
ttpd from getattr access on the file /var/www/html/test42.txt. For complete \
SELinux messages. run sealert -l b2a84386-54c1-4344-96fb-dcf969776696
Apr 12 21:44:16  centos65  setroubleshoot: SELinux is preventing /usr/sbin/h\
ttpd from getattr access on the file /var/www/html/test42.txt. For complete \
SELinux messages. run sealert -l b2a84386-54c1-4344-96fb-dcf969776696
```

So we run the command it suggests...

```
[root@linux ~]# sealert -l b2a84386-54c1-4344-96fb-dcf969776696
SELinux is preventing /usr/sbin/httpd from getattr access on the file /va\
r/www/html/test42.txt.

*****  Plugin restorecon (92.2 confidence) suggests  **********************

If you want to fix the label.
/var/www/html/test42.txt default label should be httpd_sys_content_t.
Then you can run restorecon.
Do
# /sbin/restorecon -v /var/www/html/test42.txt
...
```

We follow the friendly advice and try again to download our file:

```
[root@linux ~]# /sbin/restorecon -v /var/www/html/test42.txt
/sbin/restorecon reset /var/www/html/test42.txt context unconfined_u:objec\
t_r:samba_share_t:s0->unconfined_u:object_r:httpd_sys_content_t:s0
[root@linux ~]# wget http://localhost/test42.txt
--2014-04-12 21:54:03--  http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
```

It works!

23.23. booleans

Booleans are on/off switches

```
[root@linux ~]# getsebool -a | head
abrt_anon_write --> off
abrt_handle_event --> off
allow_console_login --> on
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
allow_daemons_use_tcp_wrapper --> off
allow_daemons_use_tty --> on
allow_domain_fd_use --> on
allow_execheap --> off
allow_execmem --> on
```

You can set and read individual booleans.

```
[root@linux ~]# setsebool httpd_read_user_content=1
[root@linux ~]# getsebool httpd_read_user_content
httpd_read_user_content --> on
[root@linux ~]# setsebool httpd_enable_homedirs=1
[root@linux ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

You can set these booleans permanent.

```
[root@linux ~]# setsebool -P httpd_enable_homedirs=1
[root@linux ~]# setsebool -P httpd_read_user_content=1
```

The above commands regenerate the complete /etc/selinux/targeted directory!

```
[root@linux ~]# cat /etc/selinux/targeted/modules/active/booleans.local
# This file is auto-generated by libsemanage
# Do not edit directly.

httpd_enable_homedirs=1
httpd_read_user_content=1
```

Part VII.

Scripting 103

24. reproducible virtual environments with Vagrant

(Written by Bert Van Vreckem, https://github.com/bertvv)

Vagrant is a tool that allows you to define an environment consisting of virtual machines that is entirely reproducible. It was originally developed by Mitchell Hashimoto and is now maintained by HashiCorp.

Vagrant is a command-line tool that talks to a hypervisor, such as VirtualBox, to create and manage virtual machines. It uses a configuration file called `Vagrantfile` to define the VMs and their settings. The `Vagrantfile` is written in Ruby, but you don't need to know Ruby to use Vagrant. One of the authors of this course has prepared a Vagrantfile that parses a YAML file to define the VMs and their settings, so you don't need to learn Ruby in order to be productive with Vagrant.

In order to create a VM, Vagrant will first download an initial VM image, called *base box* in Vagrant terminology, and then it will start the VM and configure it according to the settings in the `Vagrantfile`. This process is called *provisioning*.

The user can write a provisioning script in Bash or another infrastructure automation tool like Ansible, which will be executed inside the VM to install and configure software.

The entire definition of VMs is then comprised of the `Vagrantfile` and the provisioning scripts. This makes it easy to share the environment with other developers, and to version control the environment settings. It also makes it easy to recreate the environment on another machine, or to recreate the environment from scratch after it has been destroyed.

Vagrant is a great tool for developers who need to test their software in a local environment that is set up to resemble the acceptance/production environment. It is also useful for system administrators who need to test their infrastructure automation scripts locally before deploying on production infrastructure.

Learning goals:

- Installing Vagrant and creating a VirtualBox VM with it
- Knowing and using the basic Vagrant commands to manage VMs
- Writing a provisioning script in Bash to configure the VM

24.1. install Vagrant and scaffolding code

Vagrant can be installed on Windows, macOS, and Linux. The installation instructions are available on the Vagrant website. For example, on Windows, you can install Vagrant by opening an elevated PowerShell prompt and running the following command:

```
> winget install Hashicorp.Vagrant
```

Users of macOS can install Vagrant using Homebrew and Linux users can install Vagrant using their package manager. Do remark that some Linux distributions may have a version of Vagrant that does not work with VirtualBox out-of-the-box, but with the native libvirt hypervisor. In that case, it's best to follow the instructions on the Vagrant website and ensure you install a package provided by Hashicorp.

Remark that Vagrant is a command line application, so you will need to open a terminal to use it. Regardless of your operating system, the default terminal will do, e.g. Powershell for Windows, Bash or zsh for macOS or Bash for Linux. In fact, after you ascertain that Vagrant is installed correctly, you don't even need to start the VirtualBox GUI anymore!

Check that the installation was successful by running `vagrant --version` in your terminal.

```
> vagrant --version
Vagrant 2.4.1
```

The following step is to create some scaffolding code for a Vagrant project. You could create a new directory go to that directory and issue the command `vagrant init`. A `Vagrantfile` will be created that you can then edit to define the VMs. However, one of the authors of this course has prepared some starter code that you can download from the Github repository bertvv/vagrant-shell-skeleton. You can download the code as a zip file and extract it to a directory of your choice. The directory will contain a `Vagrantfile`, a file called `vagrant-hosts.yml` that contains the list of VMs to be created, and a directory called `provisioning` that contains the provisioning scripts.

24.2. Creating and booting a VM

In the console transcripts below, do remark that the commands are executed on your physical system, not inside a VM. We'll indicate these commands by prefixing them with the character >. Commands issued inside the VM will be prefixed with the prompt that you will see there, e.g. `vagrant@srv001:~$`.

We assume that you extracted the zip with the starter code to a directory called `vagrant-demo`. Change to that directory and issue the command `vagrant status`.

```
> cd vagrant-demo
> vagrant status
Current machine states:

srv001                    not created (virtualbox)

The environment has not yet been created. Run `vagrant up` to
create the environment. If a machine is not created, only the
default provider will be shown. So if a provider is not listed,
then the machine is not created for that environment.
```

So, apparently, this environment consists of a single VM called `srv001` that has not been created yet. Let's create it by running `vagrant up`. You may want to start the VirtualBox GUI to see the effect of the Vagrant commands, but as mentioned before, this is not necessary.

There's a lot going on if you do this for the first time, so take a look at the output. This is what you may expect:

```
> vagrant up
Bringing machine 'srv001' up with 'virtualbox' provider...
==> srv001: Box 'bento/almalinux-9' could not be found. Attempting to find and install...
    srv001: Box Provider: virtualbox
    srv001: Box Version: >= 0
==> srv001: Loading metadata for box 'bento/almalinux-9'
    srv001: URL: https://vagrantcloud.com/api/v2/vagrant/bento/almalinux-9
==> srv001: Adding box 'bento/almalinux-9' (v202401.31.0) for provider: virtualbox (amd64)
      srv001: Downloading: https://vagrantcloud.com/bento/boxes/almalinux-
9/versions/202401.31.0/providers/virtualbox/amd64/vagrant.box
```

```
    srv001:
==> srv001: Successfully added box 'bento/almalinux-9' (v202401.31.0) for 'virtualbox (amd6
==> srv001: Importing base box 'bento/almalinux-9'...
==> srv001: Matching MAC address for NAT networking...
==> srv001: Checking if box 'bento/almalinux-9' version '202401.31.0' is up to date...
==> srv001: Setting the name of the VM: vagrant-demo_srv001_1708772855256_55931
==> srv001: Clearing any previously set network interfaces...
==> srv001: Preparing network interfaces based on configuration...
    srv001: Adapter 1: nat
    srv001: Adapter 2: hostonly
==> srv001: Forwarding ports...
    srv001: 22 (guest) => 2222 (host) (adapter 1)
==> srv001: Running 'pre-boot' VM customizations...
==> srv001: Booting VM...
==> srv001: Waiting for machine to boot. This may take a few minutes...
    srv001: SSH address: 127.0.0.1:2222
    srv001: SSH username: vagrant
    srv001: SSH auth method: private key
    srv001:
    srv001: Vagrant insecure key detected. Vagrant will automatically replace
    srv001: this with a newly generated keypair for better security.
    srv001:
    srv001: Inserting generated public key within guest...
==> srv001: Machine booted and ready!
==> srv001: Checking for guest additions in VM...
==> srv001: Setting hostname...
==> srv001: Configuring and enabling network interfaces...
==> srv001: Mounting shared folders...
    srv001: /vagrant => C:/Users/student/Downloads/vagrant-demo
==> srv001: Running provisioner: shell...
   srv001: Running: C:/Users/student/AppData/Local/Temp/vagrant-shell20240224-
2812-261dwi.sh
    srv001: [LOG]  Starting common provisioning tasks
    srv001: [LOG]  Starting server specific provisioning tasks on srv001
```

Let's go through this step by step:

- First, Vagrant checks if the base box `bento/almalinux-9` is available locally. If not, it will download it from the Vagrant Cloud. This is a public repository of Vagrant boxes that you can use to create VMs. The box is a pre-configured VM image that Vagrant uses as a starting point to create a VM. The box is downloaded only once and stored on your system. If you want to create another VM based on the same box, Vagrant will use the local copy.
- The line with `Importing base box 'bento/almalinux-9'` indicates that the box is being imported into VirtualBox. That means that a VirtualBox VM is created, the disk image of the base box is copied to the directory of that VM, the VM is configured according to the settings in the `Vagrantfile` (memory, CPU, network interfaces, etc.).
- The line with `Booting VM...` indicates that the VM is started. Vagrant uses SSH to communicate with the VM, so it needs to wait until the SSH server inside the VM is ready to accept connections. This may take a while. Vagrant base boxes should always have a default user `vagrant` with password `vagrant` and a private key that Vagrant uses to authenticate to the VM. Vagrant will replace this insecure key with a newly generated keypair for better security.
- Vagrant then applies some basic settings, like the hostname and network interfaces.
- Next, on the line with `Mounting shared folders...` , Vagrant will make the directory where the `Vagrantfile` is located available inside the VM. This is very useful to share files between the host and the VM. The directory is mounted at `/vagrant` inside the VM.

- Finally, Vagrant will run the provisioning script. In this setup, the provisioning script should be located inside the directory `provisioning` and should have the same name as the VM (in this case `srv001.sh`). The script is executed inside the VM with root privileges and should install and configure the software that is needed for the VM to fulfill its role. In this case, the script will only print some log messages.

24.3. Basic Vagrant commands

After the VM has been created, you can check its status with the command `vagrant status`. You can also see the VM in the VirtualBox GUI. The VM is running and you can connect to it with SSH. Vagrant provides a command to do this: `vagrant ssh`. This command will open an SSH connection to the VM and log in as the user `vagrant`.

```
> vagrant status
Current machine states:

srv001                    running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
> vagrant ssh srv001

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
[vagrant@srv001 ~]$ pwd
/home/vagrant
[vagrant@srv001 ~]$ ls /vagrant
LICENSE.md  provisioning  README.md  Vagrantfile  vagrant-hosts.yml
[vagrant@srv001 ~]$
```

The user Vagrant has sudo privileges, so you can execute commands as root.

The table below lists the most important Vagrant commands. You can run these commands from the directory where the `Vagrantfile` is located.

Command	Description
`vagrant up`	Create and start the VMs defined in the `Vagrantfile`
`vagrant halt`	Stop the VMs
`vagrant suspend`	Suspend the VMs (save the state to disk and stop the VM)
`vagrant resume`	Resume the VMs (start the VMs from the saved state)
`vagrant reload`	Reload the VMs (restart the VMs)
`vagrant destroy`	Destroy the VMs (delete the VMs and their disk images)
`vagrant status`	Show the status of the VMs in the current environment
`vagrant global-status`	Show the status of all Vagrant VMs on your system
`vagrant ssh`	Open an SSH connection to the VM
`vagrant provision`	Run the provisioning script

Add the name of a VM to apply the command to that specific VM, e.g. `vagrant halt srv001`.

24.4. Updating the provisioning script

The provisioning script `provisioning/srv001.sh` is a Bash script that is executed as the final step of `vagrant up`, when creating the VM. If you reload the VM or boot it back up at a later time, the provisioning script will not be executed again. If you want to re-run the provisioning script, you can use the command `vagrant provision`. This is useful if you have made changes to the provisioning script and want to apply them to the VM.

At this time, the script does nothing useful. Let's change that. Open the file `provisioning/srv001.sh` in a text editor, either on your physical system, or inside the VM (under directory /vagrant). Add the following lines to the end of the file:

```
dnf -y install httpd
systemctl enable --now httpd
```

Run the provisioning script again with the command `vagrant provision srv001`. This will install the Apache web server and start it. You can check if the web server is running by executing the command `curl localhost` inside the VM. You should see the HTML code for the default web page of the Apache web server.

You can make changes to the provisioning script and run it again with `vagrant provision`. There's two important things to keep in mind when you write a provisioning script:

- The script should be able to run **non-interactively**. This means that it should *never* ask for user input. If it does, the script will fail immediately and Vagrant will issue an error message. Remark that the `dnf` command in the example above uses the `-y` option to answer "yes" to all questions so that it can run non-interactively.
- The script should be **idempotent**. This means that it should be safe to run the script multiple times, e.g. when you update the script and run it again. If a desired operation was already performed, the script should not try it again. Some Linux commands are inherently idempotent, like `dnf install`, but others are not. For example, if your script wants to add a user with the `useradd` command, running it the second time wil fail. In this case, you should first test if the user exists, and only run the `useradd` command if they don't.

24.5. Changing VM settings, adding VMs

In this setup, the VM settings can be changed in the file `vagrant-hosts.yml`. This file is a YAML file that contains a list of VMs and their settings. The file looks like this:

```
---
- name: srv001
  box: bento/almalinux-9
  ip: 192.168.56.31
```

Every VM should at least have a `name` defined, other settings are optional.

The `box` setting determines the base box that will be used to create the VM. You can discover available boxes on the Vagrant Cloud. We used an AlmaLinux 9 box created by the Bento project.

The advantage of setting an `ip` address is that your VM can be reached over the network from your physical system with a predictable IP address. Note that the network range 192.169.56.0/24 is used by VirtualBox for a host-only network. The first host-only network that you create in VirtualBox will have this exact range. Your physical system wil get the IP address 192.168.56.1 and your VM the one indicated above. Check this by pinging the VM from your physical system and vice versa!

```
> ping 192.168.56.31

Pinging 192.168.56.31 with 32 bytes of data:
Reply from 192.168.56.31: bytes=32 time=3ms TTL=64
Reply from 192.168.56.31: bytes=32 time<1ms TTL=64
Reply from 192.168.56.31: bytes=32 time<1ms TTL=64
Reply from 192.168.56.31: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.31:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 0ms
> vagrant ssh srv001
[vagrant@srv001 ~]$ ping -c3 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=128 time=0.903 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=128 time=2.05 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=128 time=0.725 ms

--- 192.168.56.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.725/1.226/2.050/0.587 ms
```

Now you know this, you can check if the web server can be reached from your physical system by opening a web browser and navigating to http://192.168.56.31. You should see the default web page of the Apache web server.

If you want to add another VM, you can add another entry to the `vagrant-hosts.yml` file. For example:

```yaml
---
- name: srv001
  box: bento/almalinux-9
  ip: 192.168.56.31

- name: srv002
  box: bento/almalinux-9
  ip: 192.168.56.32
  memory: 2048
  cpus: 2
```

This second VM uses the same base box, but we assigned an extra processor core and more memory.

Save the changes to the file and run `vagrant status`:

```
> vagrant status
Current machine states:

srv001                    running (virtualbox)
srv002                    not created (virtualbox)
```

Start the new VM with `vagrant up srv002`. You can then SSH into the VM with `vagrant ssh srv002`. You can also run the provisioning script with `vagrant provision srv002`. Remark that at this time there is no provisioning script for `srv002` yet! Copy the `srv001.sh` script to `provisioning/srv002.sh` and modify it to your liking.

When you have multiplte VMs, you will probably repeat some steps for all of them. For this purpose, the starter code already provides two extra scripts:

- `provisioning/common.sh` contains tasks that are common to all VMs, like installing a package that is needed on all VMs (e.g. `bash-completion`, `vim-enhanced`, ...), ensuring the firewall is running, etc.
- `provisioning/utils.sh` contains reusible functions that can be called from your provisioning script. As an example, it has functions for printing log messages (`log`, `debug` and `error`), an idempotent function to add a user (`ensure_user_exists`), or a group (`ensure_group_exists`), etc.

A comprehensive list of all settings that you can define for a VM in `vagrant-hosts.yml`:

- Box settings
  - `box`: A box name in the form `USER/BOX` (e.g. `bento/almalinux-9`) is fetched from Vagrant Cloud.
  - `box_url`: Download the box from the specified URL instead of from Vagrant Cloud.
- VM properties
  - `memory`: The amount of memory to allocate to the VM.
  - `cpus`: The number of CPUs to allocate to the VM.
- Network settings
  - `ip`: by default, an IP will be assigned by DHCP. If you want a fixed addres, specify it.
  - `netmask`: by default, the network mask is `255.255.255.0`. If you want another one, it should be specified.
  - `mac`: The MAC address to be assigned to the NIC. Several notations are accepted, including "Linux-style" (`00:11:22:33:44:55`) and "Windows-style" `00-11-22-33-44-55`). The separator characters can be omitted altogether (`001122334455`).
  - `intnet`: If set to `true`, the network interface will be attached to an internal network rather than a host-only adapter.
  - `auto_config`: If set to `false`, Vagrant will not attempt to configure the network interface.
  - `forwarded_ports`: a list of dicts that specify port forwarding. Each dict should have a `host` and a `guest` key. The `host` key specifies the port on the host system, and the `guest` key specifies the port on the guest system.
- Login settings:
  - `ssh_username` and `ssh_password`: Credentials for logging in to the VM (if the VM does not use the default SSH key or username/password combination).
- `synced_folders`: A list of dicts that specify synced folders. `src` and `dest` are mandatory, `options:` are optional. For the possible options, see the Vagrant documentation. Keys of options should be prefixed with a colon, e.g. `:owner:`.

An elaborate example of a host definition:

```yaml
---
- name: srv002
  box: bento/fedora-latest
  memory: 2048
  cpus: 2
  ip: 172.20.0.10
  netmask: 255.255.0.0
  mac: '13:37:de:ad:be:ef'
  forwarded_ports:
    - host: 8080
      guest: 80
    - host: 8443
      guest: 443
  synced_folders:
    - src: test
```

```
      dest: /tmp/test
    - src: www
      dest: /var/www/html
      options:
        :create: true
        :owner: root
        :group: root
        :mount_options: ['dmode=0755', 'fmode=0644']
```

24.6. Managing base boxes

Vagrant base boxes are stored on your system. You can an overview of locally available base boxes with `vagrant box list`, for example:

```
> vagrant box list
bento/almalinux-9   (virtualbox, 202401.31.0, (amd64))
bento/fedora-latest (virtualbox, 202309.08.0)
bento/ubuntu-22.04  (virtualbox, 202309.08.0)
vyos/current        (virtualbox, 20231011.00.22)
```

Installing a new base box (without creating a VM) is done with the command `vagrant box add`. For example, to add the base box `bento/debian-12.4`:

```
> vagrant box add bento/debian-12.4
==> box: Loading metadata for box 'bento/debian-12.4'
    box: URL: https://vagrantcloud.com/api/v2/vagrant/bento/debian-12.4
This box can work with multiple providers! The providers that it
can work with are listed below. Please review the list and choose
the provider you will be working with.

1) parallels
2) virtualbox
3) vmware_desktop

Enter your choice: 2
==> box: Adding box 'bento/debian-12.4' (v202401.31.0) for provider: virtualbox (amd64)
          box:   Downloading:   https://vagrantcloud.com/bento/boxes/debian-
12.4/versions/202401.31.0/providers/virtualbox/amd64/vagrant.box
    box:
==> box: Successfully added box 'bento/debian-12.4' (v202401.31.0) for 'virtualbox (amd64)'
```

You can prevent Vagrant from asking for the provider by specifying it on the command line:

```
> vagrant box add --provider=virtualbox bento/debian-12.4
```

Now, if you want to create a VM based on Debian 12.4, you won't have to download the box at that point.

You can remove a base box with `vagrant box remove`. For example, to remove the base box `bento/almalinux-9`:

```
> vagrant box remove bento/almalinux-9
Removing box 'bento/almalinux-9' (v202401.31.0) with provider 'virtualbox' ...
```

Remark that we still have a few VMs based on this box. If we need to recreate these VMs later on, Vagrant will have to download the box again.

Every time you run `vagrant up`, Vagrant will check if the base box is still up-to-date. If a new version of the box is available, Vagrant issue a warning. In that case, you can update all base boxes used in the current environment with `vagrant box update`.

Updating a specific base box (not tied to the current environment) can be done by specifying the box name, e.g:

```
> vagrant box update --box bento/fedora-latest
Checking for updates to 'bento/fedora-latest'
Latest installed version: 202309.08.0
Version constraints: > 202309.08.0
Provider: virtualbox
Updating 'bento/fedora-latest' with provider 'virtualbox' from version
'202309.08.0' to '202401.31.0'...
Loading metadata for box 'https://vagrantcloud.com/api/v2/vagrant/bento/fedora-
latest'
Adding box 'bento/fedora-latest' (v202401.31.0) for provider: virtualbox (amd64)
Downloading: https://vagrantcloud.com/bento/boxes/fedora-latest/versions/202401.31.0/prov

Successfully added box 'bento/fedora-latest' (v202401.31.0) for 'virtualbox (amd64)'!
```

Now, Vagrant does keep the old version of the box

```
> vagrant box list
...
bento/fedora-latest (virtualbox, 202309.08.0)
bento/fedora-latest (virtualbox, 202401.31.0, (amd64))
...
```

You can remove the old version with by explicity specifying it:

```
> vagrant box remove --box-version=202309.08.0 bento/fedora-latest
Removing box 'bento/fedora-latest' (v202309.08.0) with provider 'virtualbox'...
```

24.7. Exercises

1. Start with the scaffolding code of https://github.com/bertvv/vagrant-shell-skeleton.
2. Add entries to `vagrant-hosts.yml` for four VMs: `debian`, `ubuntu`, `fedora` and `alma` with the latest available base boxes from the Bento project for each distro. Assign them with an IP address in the network 192.168.56.0/24 (the default host-only network for VirtualBox).
3. Add install scripts to the `provisioning/` folder for each.

You now have a multi-VM environment to experiment with the different Linux distributions. You can use this environment to practice the commands and concepts you learn in the other chapters of this book. If the VM is broken, you can easily destroy it and recreate it from scratch! If you want to experiment with other distros, feel free to add more VMs to your environment. ## Solutions

1. Download the code by clicking the green button “Code” on the GitHub page and selecting “Download ZIP”. Unpack the ZIP file in a directory of your choice and rename it to e.g. `vagrant-demo-env` (or any other name you like).

   Open a terminal in this directory and run `git init .` to initialize a new Git repository. This is not strictly necessary, but it’s a good habit to keep your work under version control. You can now add the files to the repository with `git add .` and commit them with `git commit -m "Initial commit"`.

2. Open `vagrant-hosts.yml` and add the following content. Check the latest versions of the bae boxes on https://app.vagrantup.com/bento and update the version numbers if necessary!

```yaml
---
- name: alma
  box: bento/almalinux-9
  ip: 192.168.56.11

- name: fedora
  box: bento/fedora-latest
  ip: 192.168.56.12

- name: debian
  box: bento/debian-12.4
  ip: 192.168.56.21

- name: ubuntu
  box: bento/ubuntu-22.04
  ip: 192.168.56.22
```

3. Copy the file `srv001.sh` to `alma.sh`, `fedora.sh`, etc. and add any commands for installing packages or configuring the system.

   Don’t forget to add the new files to the Git repository and commit your changes.

Part VIII.

Scripting 201; job scheduling

25. more scripting

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

25.1. eval

`eval` reads arguments as input to the shell (the resulting commands are executed). This allows using the value of a variable as a variable.

```
student@linux:~/test42$ answer=42
student@linux:~/test42$ word=answer
student@linux:~/test42$ eval x=\$$word ; echo $x
42
```

Both in `bash` and `Korn` the arguments can be quoted.

```
kahlan@solexp11$ answer=42
kahlan@solexp11$ word=answer
kahlan@solexp11$ eval "y=\$$word" ; echo $y
42
```

Sometimes the `eval` is needed to have correct parsing of arguments. Consider this example where the `date` command receives one parameter `1 week ago`.

```
student@linux~$ date --date="1 week ago"
Thu Mar  8 21:36:25 CET 2012
```

When we set this command in a variable, then executing that variable fails unless we use `eval`.

```
student@linux~$ lastweek='date --date="1 week ago"'
student@linux~$ $lastweek
date: extra operand `ago"'
Try `date --help' for more information.
student@linux~$ eval $lastweek
Thu Mar  8 21:36:39 CET 2012
```

25.2. (( ))

The `(( ))` allows for evaluation of numerical expressions.

```
student@linux:~/test42$ (( 42 > 33 )) && echo true || echo false
true
student@linux:~/test42$ (( 42 > 1201 )) && echo true || echo false
false
student@linux:~/test42$ var42=42
student@linux:~/test42$ (( 42 == var42 )) && echo true || echo false
true
student@linux:~/test42$ (( 42 == $var42 )) && echo true || echo false
true
student@linux:~/test42$ var42=33
student@linux:~/test42$ (( 42 == var42 )) && echo true || echo false
false
```

25.3. let

The `let` built-in shell function instructs the shell to perform an evaluation of arithmetic expressions. It will return 0 unless the last arithmetic expression evaluates to 0.

```
[student@linux ~]$ let x="3 + 4" ; echo $x
7
[student@linux ~]$ let x="10 + 100/10" ; echo $x
20
[student@linux ~]$ let x="10-2+100/10" ; echo $x
18
[student@linux ~]$ let x="10*2+100/10" ; echo $x
30
```

The `shell` can also convert between different bases.

```
[student@linux ~]$ let x="0×FF" ; echo $x
255
[student@linux ~]$ let x="0×C0" ; echo $x
192
[student@linux ~]$ let x="0×A8" ; echo $x
168
[student@linux ~]$ let x="8#70" ; echo $x
56
[student@linux ~]$ let x="8#77" ; echo $x
63
[student@linux ~]$ let x="16#c0" ; echo $x
192
```

There is a difference between assigning a variable directly, or using `let` to evaluate the arithmetic expressions (even if it is just assigning a value).

```
kahlan@solexp11$ dec=15 ; oct=017 ; hex=0×0f
kahlan@solexp11$ echo $dec $oct $hex
15 017 0×0f
kahlan@solexp11$ let dec=15 ; let oct=017 ; let hex=0×0f
kahlan@solexp11$ echo $dec $oct $hex
15 15 15
```

25.4. case

You can sometimes simplify nested if statements with a `case` construct.

```
[student@linux ~]$ ./help
What animal did you see ? lion
You better start running fast!
[student@linux ~]$ ./help
What animal did you see ? dog
Don't worry, give it a cookie.
[student@linux ~]$ cat help
#!/bin/bash
#
# Wild Animals Helpdesk Advice
#
echo -n "What animal did you see ? "
read animal
case $animal in
        "lion" | "tiger")
                echo "You better start running fast!"
        ;;
        "cat")
                echo "Let that mouse go ... "
        ;;
        "dog")
                echo "Don't worry, give it a cookie."
        ;;
        "chicken" | "goose" | "duck" )
                echo "Eggs for breakfast!"
        ;;
        "liger")
                echo "Approach and say 'Ah you big fluffy kitty ... '."
        ;;
        "babelfish")
                echo "Did it fall out your ear ?"
        ;;
        *)
                echo "You discovered an unknown animal, name it!"
        ;;
esac
[student@linux ~]$
```

25.5. shell functions

Shell `functions` can be used to group commands in a logical way.

```
kahlan@solexp11$ cat funcs.ksh
#!/bin/ksh

function greetings {
echo Hello World!
echo and hello to $USER to!
}
```

```
echo We will now call a function
greetings
echo The end
```

This is sample output from this script with a `function`.

```
kahlan@solexp11$ ./funcs.ksh
We will now call a function
Hello World!
and hello to kahlan to!
The end
```

A shell function can also receive parameters.

```
kahlan@solexp11$ cat addfunc.ksh
#!/bin/ksh

function plus {
let result="$1 + $2"
echo  $1 + $2 = $result
}

plus 3 10
plus 20 13
plus 20 22
```

This script produces the following output.

```
kahlan@solexp11$ ./addfunc.ksh
3 + 10 = 13
20 + 13 = 33
20 + 22 = 42
```

25.6. practice : more scripting

1. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:       5 + 2 = 7
Product:   5 x 2 = 10
```

2. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

3. Improve the previous script to congratulate the user if the sum equals the product.

4. Write a script with a case insensitive case statement, using the shopt nocasematch option. The nocasematch option is reset to the value it had before the scripts started.

5. If time permits (or if you are waiting for other students to finish this practice), take a look at Linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

25.7. solution : more scripting

1. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:       5 + 2 = 7
Product:   5 x 2 = 10
```

```
#!/bin/bash

echo -n "Enter a number : "
read n1

echo -n "Enter another number : "
read n2

let sum="$n1+$n2"
let pro="$n1*$n2"

echo -e "Sum\t: $n1 + $n2 = $sum"
echo -e "Product\t: $n1 * $n2 = $pro"
```

2. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

```
echo -n "Enter a number between 1 and 100 : "
read n1

if [ $n1 -lt 1 -o $n1 -gt 100 ]
then
       echo Wrong number...
       exit 1
fi
```

3. Improve the previous script to congratulate the user if the sum equals the product.

```
if [ $sum -eq $pro ]
then echo Congratulations $sum == $pro
fi
```

4. Write a script with a case insensitive case statement, using the shopt nocasematch option. The nocasematch option is reset to the value it had before the scripts started.

```
#!/bin/bash
#
# Wild Animals Case Insensitive Helpdesk Advice
#

if shopt -q nocasematch; then
  nocase=yes;
else
  nocase=no;
```

```
  shopt -s nocasematch;
fi

echo -n "What animal did you see ? "
read animal

case $animal in
        "lion" | "tiger")
                echo "You better start running fast!"
        ;;
        "cat")
                echo "Let that mouse go ... "
        ;;
        "dog")
                echo "Don't worry, give it a cookie."
        ;;
        "chicken" | "goose" | "duck" )
                echo "Eggs for breakfast!"
        ;;
        "liger")
                echo "Approach and say 'Ah you big fluffy kitty.'"
        ;;
        "babelfish")
                echo "Did it fall out your ear ?"
        ;;
        *)
                echo "You discovered an unknown animal, name it!"
        ;;
esac

if [ nocase = yes ] ; then
        shopt -s nocasematch;
else
        shopt -u nocasematch;
fi
```

5. If time permits (or if you are waiting for other students to finish this practice), take a look at Linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

26. background jobs

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

26.1. background processes

26.1.1. jobs

Stuff that runs in background of your current shell can be displayed with the `jobs` command. By default you will not have any `jobs` running in background.

```
root@linux ~# jobs
root@linux ~#
```

This `jobs` command will be used several times in this section.

26.1.2. control-Z

Some processes can be `suspended` with the `Ctrl-Z` key combination. This sends a `SIGSTOP` signal to the `Linux kernel`, effectively freezing the operation of the process.

When doing this in `vi(m)`, then `vi(m)` goes to the background. The background `vi(m)` can be seen with the `jobs` command.

```
[student@linux ~]$ vi procdemo.txt

[5]+  Stopped                 vim procdemo.txt
[student@linux ~]$ jobs
[5]+  Stopped                 vim procdemo.txt
```

26.1.3. & ampersand

Processes that are started in background using the `&` character at the end of the command line are also visible with the `jobs` command.

```
[student@linux ~]$ find / > allfiles.txt 2> /dev/null &
[6] 5230
[student@linux ~]$ jobs
[5]+  Stopped                 vim procdemo.txt
[6]-  Running                 find / >allfiles.txt 2>/dev/null &
[student@linux ~]$
```

26.1.4. jobs -p

An interesting option is `jobs -p` to see the `process id` of background processes.

```
[student@linux ~]$ sleep 500 &
[1] 4902
[student@linux ~]$ sleep 400 &
[2] 4903
[student@linux ~]$ jobs -p
4902
4903
[student@linux ~]$ ps `jobs -p`
  PID TTY      STAT   TIME COMMAND
 4902 pts/0    S      0:00 sleep 500
 4903 pts/0    S      0:00 sleep 400
[student@linux ~]$
```

26.1.5. fg

Running the `fg` command will bring a background job to the foreground. The number of the background job to bring forward is the parameter of `fg`.

```
[student@linux ~]$ jobs
[1]   Running                 sleep 1000 &
[2]-  Running                 sleep 1000 &
[3]+  Running                 sleep 2000 &
[student@linux ~]$ fg 3
sleep 2000
```

26.1.6. bg

Jobs that are `suspended` in background can be started in background with `bg`. The `bg` will send a `SIGCONT` signal.

Below an example of the sleep command (suspended with `Ctrl-Z`) being reactivated in background with `bg`.

```
[student@linux ~]$ jobs
[student@linux ~]$ sleep 5000 &
[1] 6702
[student@linux ~]$ sleep 3000

[2]+  Stopped                 sleep 3000
[student@linux ~]$ jobs
[1]-  Running                 sleep 5000 &
[2]+  Stopped                 sleep 3000
[student@linux ~]$ bg 2
[2]+ sleep 3000 &
[student@linux ~]$ jobs
[1]-  Running                 sleep 5000 &
[2]+  Running                 sleep 3000 &
[student@linux ~]$
```

26.2. practice : background processes

1. Use the `jobs` command to verify whether you have any processes running in background.

2. Use `vi` to create a little text file. Suspend `vi` in background.

3. Verify with `jobs` that `vi` is suspended in background.

4. Start `find / > allfiles.txt 2>/dev/null` in foreground. Suspend it in background before it finishes.

5. Start two long `sleep` processes in background.

6. Display all `jobs` in background.

7. Use the `kill` command to suspend the last `sleep` process.

8. Continue the `find` process in background (make sure it runs again).

9. Put one of the `sleep` commands back in foreground.

10. (if time permits, a general review question...) Explain in detail where the numbers come from in the next screenshot. When are the variables replaced by their value ? By which shell ?

```
[student@linux ~]$ echo $$ $PPID
4224 4223
[student@linux ~]$ bash -c "echo $$ $PPID"
4224 4223
[student@linux ~]$ bash -c 'echo $$ $PPID'
5059 4224
[student@linux ~]$ bash -c `echo $$ $PPID`
4223: 4224: command not found
```

26.3. solution : background processes

1. Use the `jobs` command to verify whether you have any processes running in background.

```
jobs (maybe the catfun is still running?)
```

2. Use `vi` to create a little text file. Suspend `vi` in background.

```
vi text.txt
(inside vi press ctrl-z)
```

3. Verify with `jobs` that `vi` is suspended in background.

```
[student@linux ~]$ jobs
[1]+  Stopped                 vim text.txt
```

4. Start `find / > allfiles.txt 2>/dev/null` in foreground. Suspend it in background before it finishes.

```
[student@linux ~]$ find / > allfiles.txt 2>/dev/null
   (press ctrl-z)
[2]+  Stopped                 find / > allfiles.txt 2> /dev/null
```

5. Start two long `sleep` processes in background.

```
sleep 4000 & ; sleep 5000 &
```

6. Display all `jobs` in background.

```
[student@linux ~]$ jobs
[1]-  Stopped                 vim text.txt
[2]+  Stopped                 find / > allfiles.txt 2> /dev/null
[3]   Running                 sleep 4000 &
[4]   Running                 sleep 5000 &
```

7. Use the `kill` command to suspend the last `sleep` process.

```
[student@linux ~]$ kill -SIGSTOP 4519
[student@linux ~]$ jobs
[1]   Stopped                 vim text.txt
[2]-  Stopped                 find / > allfiles.txt 2> /dev/null
[3]   Running                 sleep 4000 &
[4]+  Stopped                 sleep 5000
```

8. Continue the `find` process in background (make sure it runs again).

```
bg 2 (verify the job-id in your jobs list)
```

9. Put one of the `sleep` commands back in foreground.

```
fg 3 (again verify your job-id)
```

10. (if time permits, a general review question...) Explain in detail where the numbers come from in the next screenshot. When are the variables replaced by their value ? By which shell ?

```
[student@linux ~]$ echo $$ $PPID
4224 4223
[student@linux ~]$ bash -c "echo $$ $PPID"
4224 4223
[student@linux ~]$ bash -c 'echo $$ $PPID'
5059 4224
[student@linux ~]$ bash -c `echo $$ $PPID`
4223: 4224: command not found
```

The current bash shell will replace the $$ and $PPID while scanning the line, and before executing the echo command.

```
[student@linux ~]$ echo $$ $PPID
4224 4223
```

The variables are now double quoted, but the current bash shell will replace $$ and $PPID while scanning the line, and before executing the `bash -c` command.

```
[student@linux ~]$ bash -c "echo $$ $PPID"
4224 4223
```

The variables are now single quoted. The current bash shell will `not` replace the $$ and the $PPID. The bash -c command will be executed before the variables replaced with their value. This latter bash is the one replacing the $$ and $PPID with their value.

```
[student@linux ~]$ bash -c 'echo $$ $PPID'
5059 4224
```

With backticks the shell will still replace both variable before the embedded echo is executed. The result of this echo is the two process id's. These are given as commands to bash -c. But two numbers are not commands!

```
[student@linux ~]$ bash -c `echo $$ $PPID`
4223: 4224: command not found
```

27. scheduling

(Written by Paul Cobbaut, https://github.com/paulcobbaut/)

Linux administrators use the `at` to schedule one time jobs. Recurring jobs are better scheduled with `cron`. The next two sections will discuss both tools.

27.1. one time jobs with at

27.1.1. at

Simple scheduling can be done with the `at` command. This screenshot shows the scheduling of the date command at 22:01 and the sleep command at 22:03.

```
root@linux:~# at 22:01
at> date
at> <EOT>
job 1 at Wed Aug  1 22:01:00 2007
root@linux:~# at 22:03
at> sleep 10
at> <EOT>
job 2 at Wed Aug  1 22:03:00 2007
root@linux:~#
```

In real life you will hopefully be scheduling more useful commands ;-)

27.1.2. atq

It is easy to check when jobs are scheduled with the `atq` or `at -l` commands.

```
root@linux:~# atq
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@linux:~# at -l
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@linux:~#
```

The at command understands English words like tomorrow and teatime to schedule commands the next day and at four in the afternoon.

```
root@linux:~# at 10:05 tomorrow
at> sleep 100
at> <EOT>
job 5 at Thu Aug  2 10:05:00 2007
root@linux:~# at teatime tomorrow
at> tea
```

```
at> <EOT>
job 6 at Thu Aug  2 16:00:00 2007
root@linux:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@linux:~#
```

27.1.3. atrm

Jobs in the at queue can be removed with `atrm`.

```
root@linux:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@linux:~# atrm 5
root@linux:~# atq
6       Thu Aug  2 16:00:00 2007 a root
root@linux:~#
```

27.1.4. at.allow and at.deny

You can also use the `/etc/at.allow` and `/etc/at.deny` files to manage who can schedule jobs with at.

The `/etc/at.allow` file can contain a list of users that are allowed to schedule `at` jobs. When `/etc/at.allow` does not exist, then everyone can use `at` unless their username is listed in `/etc/at.deny`.

If none of these files exist, then everyone can use `at`.

27.2. cron

27.2.1. crontab file

The `crontab(1)` command can be used to maintain the `crontab(5)` file. Each user can have their own crontab file to schedule jobs at a specific time. This time can be specified with five fields in this order: minute, hour, day of the month, month and day of the week. If a field contains an asterisk (*), then this means all values of that field.

The following example means : run script42 eight minutes after two, every day of the month, every month and every day of the week.

```
8 14 * * * script42
```

Run script8472 every month on the first of the month at 25 past midnight.

```
25 0 1 * * script8472
```

Run this script33 every two minutes on Sunday (both 0 and 7 refer to Sunday).

```
*/2 * * * 0
```

Instead of these five fields, you can also type one of these: @reboot, @yearly or @annually, @monthly, @weekly, @daily or @midnight, and @hourly.

27.2.2. crontab command

Users should not edit the crontab file directly, instead they should type `crontab -e` which will use the editor defined in the EDITOR or VISUAL environment variable. Users can display their cron table with `crontab -l`.

27.2.3. cron.allow and cron.deny

The `cron daemon crond` is reading the cron tables, taking into account the `/etc/cron.allow` and `/etc/cron.deny` files.

These files work in the same way as `at.allow` and `at.deny`. When the `cron.allow` file exists, then your username has to be in it, otherwise you cannot use `cron`. When the `cron.allow` file does not exists, then your username cannot be in the `cron.deny` file if you want to use `cron`.

27.2.4. /etc/crontab

The `/etc/crontab` file contains entries for when to run hourly/daily/weekly/monthly tasks. It will look similar to this output.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

20 3 * * *        root    run-parts --report /etc/cron.daily
40 3 * * 7        root    run-parts --report /etc/cron.weekly
55 3 1 * *        root    run-parts --report /etc/cron.monthly
```

27.2.5. /etc/cron.*

The directories shown in the next screenshot contain the tasks that are run at the times scheduled in `/etc/crontab`. The `/etc/cron.d` directory is for special cases, to schedule jobs that require finer control than hourly/daily/weekly/monthly.

```
student@linux:~$ ls -ld /etc/cron.*
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.d
drwxr-xr-x 2 root root 4096 2008-04-19 15:04 /etc/cron.daily
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.hourly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.monthly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.weekly
```

27.2.6. /etc/cron.*

Note that Red Hat uses `anacron` to schedule daily, weekly and monthly cron jobs.

```
root@linux:/etc# cat anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
```

```
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1       5       cron.daily              nice run-parts /etc/cron.daily
7       25      cron.weekly             nice run-parts /etc/cron.weekly
@monthly 45     cron.monthly            nice run-parts /etc/cron.monthly
root@linux:/etc#
```

27.3. practice : scheduling

1. Schedule two jobs with `at`, display the `at queue` and remove a job.

2. As normal user, use `crontab -e` to schedule a script to run every four minutes.

3. As root, display the `crontab` file of your normal user.

4. As the normal user again, remove your `crontab` file.

5. Take a look at the `cron` files and directories in `/etc` and understand them. What is the `run-parts` command doing ?

27.4. solution : scheduling

1. Schedule two jobs with `at`, display the `at queue` and remove a job.

```
root@linux ~# at 9pm today
at> echo go to bed >> /root/todo.txt
at> <EOT>
job 1 at 2010-11-14 21:00
root@linux ~# at 17h31 today
at> echo go to lunch >> /root/todo.txt
at> <EOT>
job 2 at 2010-11-14 17:31
root@linux ~# atq
2   2010-11-14 17:31 a root
1   2010-11-14 21:00 a root
root@linux ~# atrm 1
root@linux ~# atq
2   2010-11-14 17:31 a root
root@linux ~# date
Sun Nov 14 17:31:01 CET 2010
root@linux ~# cat /root/todo.txt
go to lunch
```

2. As normal user, use `crontab -e` to schedule a script to run every four minutes.

```
student@linux ~$ crontab -e
no crontab for paul - using an empty one
crontab: installing new crontab
```

3. As root, display the `crontab` file of your normal user.

```
root@linux ~# crontab -l -u paul
*/4 * * * * echo `date` >> /home/paul/crontest.txt
```

4. As the normal user again, remove your `crontab` file.

```
student@linux ~$ crontab -r
student@linux ~$ crontab -l
no crontab for paul
```

5. Take a look at the `cron` files and directories in `/etc` and understand them. What is the `run-parts` command doing ?

```
run-parts runs a script in a directory
```

Part IX.

SSH; troubleshooting

28. ssh client and server

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

The `secure shell` or `ssh` is a collection of tools using a secure protocol for communications with remote Linux computers.

This chapter gives an overview of the most common commands related to the use of the `sshd` server and the `ssh` client.

28.1. about ssh

28.1.1. secure shell

Avoid using `telnet`, `rlogin` and `rsh` to remotely connect to your servers. These older protocols do not encrypt the login session, which means your user id and password can be sniffed by tools like `wireshark` or `tcpdump`. To securely connect to your servers, use `ssh`.

The `ssh protocol` is secure in two ways. Firstly the connection is `encrypted` and secondly the connection is `authenticated` both ways.

An ssh connection always starts with a cryptographic handshake, followed by `encryption` of the transport layer using a symmetric cypher. In other words, the tunnel is encrypted before you start typing anything.

Then `authentication` takes place (using user id/password or public/private keys) and communication can begin over the encrypted connection.

The `ssh protocol` will remember the servers it connected to (and warn you in case something suspicious happened).

The `openssh` package is maintained by the `OpenBSD` people and is distributed with a lot of operating systems (it may even be the most popular package in the world).

28.1.2. /etc/ssh/

Configuration of `ssh` client and server is done in the `/etc/ssh` directory. In the next sections we will discuss most of the files found in `/etc/ssh/`.

28.1.3. ssh protocol versions

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

```
student@linux:/etc/ssh$ grep Protocol ssh_config
#   Protocol 2,1
student@linux:/etc/ssh$ grep Protocol sshd_config
Protocol 2
```

28.1.4. public and private keys

The ssh protocol uses the well known system of public and private keys. The below explanation is succinct, more information can be found on wikipedia.

http://en.wikipedia.org/wiki/Public-key_cryptography

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

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

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

28.1.5. rsa and dsa algorithms

This chapter does not explain the technical implementation of cryptographic algorithms, it only explains how to use the ssh tools with rsa and dsa. More information about these algorithms can be found here:

http://en.wikipedia.org/wiki/RSA_(algorithm)
http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

28.2. log on to a remote server

The following screenshot shows how to use ssh to log on to a remote computer running Linux. The local user is named paul and he is logging on as user admin42 on the remote system.

```
student@linux:~$ ssh admin42@192.168.1.30
The authenticity of host '192.168.1.30 (192.168.1.30)' can't be established.
RSA key fingerprint is b5:fb:3c:53:50:b4:ab:81:f3:cd:2e:bb:ba:44:d3:75.
Are you sure you want to continue connecting (yes/no)?
```

As you can see, the user paul is presented with an rsa authentication fingerprint from the remote system. The user can accepts this bu typing yes. We will see later that an entry will be added to the ~/.ssh/known_hosts file.

```
student@linux:~$ ssh admin42@192.168.1.30
The authenticity of host '192.168.1.30 (192.168.1.30)' can't be established.
RSA key fingerprint is b5:fb:3c:53:50:b4:ab:81:f3:cd:2e:bb:ba:44:d3:75.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.30' (RSA) to the list of known hosts.
admin42@192.168.1.30's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-26-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

1 package can be updated.
```

```
0 updates are security updates.

Last login: Wed Jun  6 19:25:57 2012 from 172.28.0.131
admin42@ubuserver:~$
```

The user can get log out of the remote server by typing `exit` or by using `Ctrl-d`.

```
admin42@ubuserver:~$ exit
logout
Connection to 192.168.1.30 closed.
student@linux:~$
```

28.3. executing a command in remote

This screenshot shows how to execute the `pwd` command on the remote server. There is no need to `exit` the server manually.

```
student@linux:~$ ssh admin42@192.168.1.30 pwd
admin42@192.168.1.30's password:
/home/admin42
student@linux:~$
```

28.4. scp

The `scp` command works just like `cp`, but allows the source and destination of the copy to be behind `ssh`. Here is an example where we copy the `/etc/hosts` file from the remote server to the home directory of user paul.

```
student@linux:~$ scp admin42@192.168.1.30:/etc/hosts /home/paul/serverhosts
admin42@192.168.1.30's password:
hosts                                         100%  809     0.8KB/s   00:00
```

Here is an example of the reverse, copying a local file to a remote server.

```
student@linux:~$ scp ~/serverhosts admin42@192.168.1.30:/etc/hosts.new
admin42@192.168.1.30's password:
serverhosts                                   100%  809     0.8KB/s   00:00
```

28.5. setting up passwordless ssh

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

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

28.5.1. ssh-keygen

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

```
[alice@linux ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alice/.ssh/id_rsa):
Created directory '/home/alice/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alice/.ssh/id_rsa.
Your public key has been saved in /home/alice/.ssh/id_rsa.pub.
The key fingerprint is:
9b:ac:ac:56:c2:98:e5:d9:18:c4:2a:51:72:bb:45:eb alice@linux
[alice@linux ~]$
```

You can use `ssh-keygen -t dsa` in the same way.

28.5.2. ~/.ssh

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

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

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

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

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

28.5.3. id_rsa and id_rsa.pub

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

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

The files will be named `id_dsa` and `id_dsa.pub` when using `dsa` instead of `rsa`.

28.5.4. copy the public key to the other computer

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

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

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

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Alice could also have used `ssh-copy-id` like in this example.

```
ssh-copy-id -i .ssh/id_rsa.pub bob@192.168.48.92
```

28.5.5. authorized_keys

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

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

28.5.6. passwordless ssh

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

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

28.6. X forwarding via ssh

Another popular feature of `ssh` is called `X11 forwarding` and is implemented with `ssh -X`.

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

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

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

28.7. troubleshooting ssh

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

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

28.8. sshd

The ssh server is called `sshd` and is provided by the `openssh-server` package.

```
root@linux~# dpkg -l openssh-server | tail -1
ii  openssh-server   1:5.9p1-5ubuntu1    secure shell (SSH) server, ...
```

28.9. sshd keys

The public keys used by the sshd server are located in `/etc/ssh` and are world readable. The private keys are only readable by root.

```
root@linux~# ls -l /etc/ssh/ssh_host_*
-rw------- 1 root root  668 Jun  7  2011 /etc/ssh/ssh_host_dsa_key
-rw-r--r-- 1 root root  598 Jun  7  2011 /etc/ssh/ssh_host_dsa_key.pub
-rw------- 1 root root 1679 Jun  7  2011 /etc/ssh/ssh_host_rsa_key
-rw-r--r-- 1 root root  390 Jun  7  2011 /etc/ssh/ssh_host_rsa_key.pub
```

28.10. ssh-agent

When generating keys with `ssh-keygen`, you have the option to enter a passphrase to protect access to the keys. To avoid having to type this passphrase every time, you can add the key to `ssh-agent` using `ssh-add`.

Most Linux distributions will start the `ssh-agent` automatically when you log on.

```
root@linux~# ps -ef | grep ssh-agent
paul     2405  2365  0 08:13 ?        00:00:00 /usr/bin/ssh-agent ...
```

This clipped screenshot shows how to use `ssh-add` to list the keys that are currently added to the `ssh-agent`

```
student@linux:~$ ssh-add -L
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAvgI+Vx5UrIsusZPl8da8URHGsxG7yivv3/\
 ...
wMGqa48Kelwom8TGb4Sgcwpp/VO/ldA5m+BGCw== student@linux
```

28.11. practice: ssh

0. Make sure that you have access to `two Linux computers`, or work together with a partner for this exercise. For this practice, we will name one of the machines the server.

1. Install `sshd` on the server

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

3. Use `ssh` to log on to the server, show your current directory and then exit the server.

4. Use `scp` to copy a file from your computer to the server.

5. Use `scp` to copy a file from the server to your computer.

6. (optional, only works when you have a graphical install of Linux) Install the xeyes package on the server and use ssh to run xeyes on the server, but display it on your client.

7. (optional, same as previous) Create a bookmark in firefox, then quit firefox on client and server. Use `ssh -X` to run firefox on your display, but on your neighbour's computer. Do you see your neighbour's bookmark ?

8. Use `ssh-keygen` to create a key pair without passphrase. Setup passwordless ssh between you and your neighbour. (or between your client and your server)

9.Verify that the permissions on the server key files are correct; world readable for the public keys and only root access for the private keys.

10. Verify that the `ssh-agent` is running.

11. (optional) Protect your keypair with a `passphrase`, then add this key to the `ssh-agent` and test your passwordless ssh to the server.

28.12. solution: ssh

0. Make sure that you have access to `two Linux computers`, or work together with a partner for this exercise. For this practice, we will name one of the machines the server.

1. Install `sshd` on the server

```
apt-get install openssh-server (on Ubuntu/Debian)
yum -y install openssh-server (on Centos/Fedora/Red Hat)
```

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

```
grep Protocol /etc/ssh/ssh*_config
```

3. Use `ssh` to log on to the server, show your current directory and then exit the server.

```
user@client$ ssh user@server-ip-address
user@server$ pwd
/home/user
user@server$ exit
```

4. Use `scp` to copy a file from your computer to the server.

```
scp localfile user@server:~
```

5. Use `scp` to copy a file from the server to your computer.

```
scp user@server:~/serverfile .
```

6. (optional, only works when you have a graphical install of Linux) Install the xeyes package on the server and use ssh to run xeyes on the server, but display it on your client.

```
on the server:
apt-get install xeyes
on the client:
ssh -X user@server-ip
xeyes
```

7. (optional, same as previous) Create a bookmark in firefox, then quit firefox on client and server. Use `ssh -X` to run firefox on your display, but on your neighbour's computer. Do you see your neighbour's bookmark ?

8. Use `ssh-keygen` to create a key pair without passphrase. Setup passwordless ssh between you and your neighbour. (or between your client and your server)

```
See solution in book "setting up passwordless ssh"
```

9. Verify that the permissions on the server key files are correct; world readable for the public keys and only root access for the private keys.

```
ls -l /etc/ssh/ssh_host_*
```

10. Verify that the `ssh-agent` is running.

```
ps fax | grep ssh-agent
```

11. (optional) Protect your keypair with a `passphrase`, then add this key to the `ssh-agent` and test your passwordless ssh to the server.

```
man ssh-keygen
man ssh-agent
man ssh-add
```

29. troubleshooting network services

(Written by Bert Van Vreckem, https://github.com/bertvv)

In this chapter, we discuss how you can troubleshoot issues while setting up network services on a Linux system. We assume that you have a basic understanding of networking concepts, that you are familiar with the Linux command line and that you are using a fairly recent Linux distribution (with `systemd`).

Learning goals:

- Applying a systematic and thorough approach to troubleshooting, using a bottom-up strategy.
- Address issues on the network acces layer: check adapter configuration and connectivity
- Address issues on the Internet layer: local network configuration, routing within the LAN
- Address issues on the transport layer: service status, used sockets and firewall settings
- Address issues on the application layer: checking log files, validate config file syntax

29.1. a bottom-up approach to troubleshooting

As you know by the time you reach this chapter, the Internet is designed as a layered system. The OSI model is often used to explain how networking works, but for our purposes, the four layer TCP/IP model is more practical and sufficient.

For reference, here is a summary of the TCP/IP model:

Layer	Protocols	Keywords
Application	HTTP, DNS, DHCP, FTP, …	
Transport	TCP, UDP	sockets, port numbers
Internet	IP, ICMP	routing, IP address
Link	Ethernet, ARP	switch, MAC address
(Physical)		cables, RF signals

The physical layer at the bottom is not really part of the TCP/IP family of protocols and is out of scope for this chapter.

In order to troubleshoot effectively, it is essential to understand that interventions on the higher layers will not solve issues if problems on the lower layers aren't fixed first. This is why you should **always use a bottom-up approach to troubleshooting**. That is, start at the bottom layer of the stack, check *everything* on that layer before moving up to the next layer.

In the following sections, we'll show what to check on each layer and which commands to use. We'll also provide some general guidelines for troubleshooting network services.

It is always a good idea to open a separate terminal window and follow the logs in real-time with `sudo journalctl -fl` or, specific to a service, `sudo journalctl -flu <service>`.

29.2. lab environment setup

In order to follow the examples provided in this chapter, you can set up a lab environment from the Github repository HoGentTIN/linux-training-labs. Fork and clone the repository or download it as a ZIP file. Follow the instructions in the main README file to get started and then the README file in the `troubleshooting/` directory.

After launching the Vagrant environment in the `troubleshooting/` directory, you will have two AlmaLinux 9 VMs running:

Host	IP	Service
`webt`	192.168.76.72	Apache web server
`dbt`	192.168.76.73	MariaDB database server

The database server is configured correctly, but the web server is not.

In the examples below, we will use these host names to refer to the systems. Remark that these systems have two adapters: `eth0` is connected to a NAT interface that provides Internet access, and `eth1` is attached to a Host-only Adapter that connects the two systems.

29.3. the link layer

On this layer, check:

- Are cables connected properly?
  - On a physical device, look for blinking lights on the network interface of the system under test and the network device it is attached to. Testing the cables with a cable tester is also a good idea.
  - On a Virtual Machine, check if the network adapter is connected to the right network.
- Are the network interfaces up?

Use the `ip link` command to check the status of the network interfaces. E.g. (with the `-br` option for a more concise output):

```
[vagrant@webt ~]$ ip -br l
lo       UNKNOWN    00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
eth0     UP         08:00:27:b3:a1:5c <BROADCAST,MULTICAST,UP,LOWER_UP>
eth1     DOWN       08:00:27:b3:29:e3 <NO-CARRIER,BROADCAST,MULTICAST,UP>
```

This system has three network interfaces:

- `lo`: the loopback interface, always up.
- `eth0`: an Ethernet adapter, up and running (indicated by `UP` and `BROADCAST`).
- `eth1`: another Ethernet adapter, not connected (indicated by `DOWN` and `NO-CARRIER`).

In fact, there is no signal on adapter `eth1`, which either indicates that the cable is unplugged, or the cable is defective. On a physical system, check the cable and ports on the system and the switch.

In VirtualBox, there's also a few things to check. Open the VM settings and go to the Network tab.

- First of all, all VMs that need to communicate with each other should be attached to the same network. In the lab setup, check the Adapter 2 tab and ensure that `webt` is connected to the same network as `dbt`. If not, make the necessary changes to `webt`. Remark that restarting the VM is not necessary. This is equivalent to plugging in a cable on another network device.
- Next, click on the "Advanced" and ensure the checkbox with label "Cable connected" is checked.

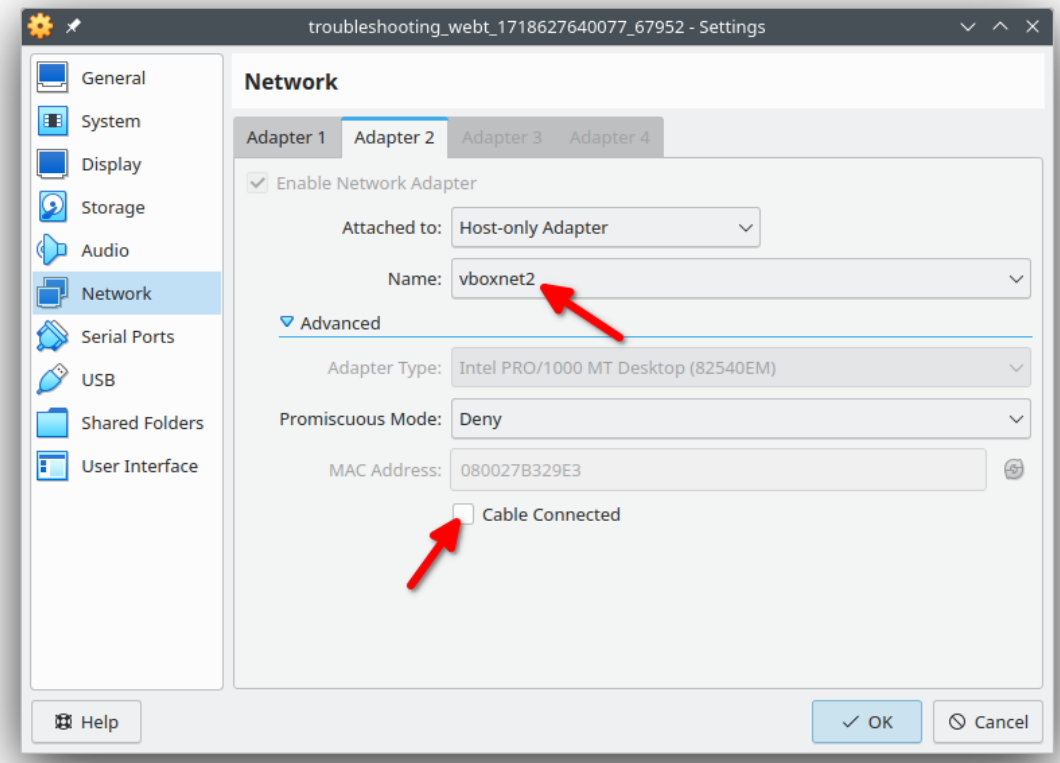


Figure 29.1.: Network settings of `webt` in VirtualBox. Adapter 2 is not attached to the correct Host-only Adapter and the cable is not connected.

After making the necessary changes, check the status of the network interfaces again:

```
[vagrant@webt ~]$ ip -br l
lo       UNKNOWN    00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
eth0     UP         08:00:27:b3:a1:5c <BROADCAST,MULTICAST,UP,LOWER_UP>
eth1     UP         08:00:27:b3:29:e3 <BROADCAST,MULTICAST,UP,LOWER_UP>
```

29.4. the internet layer

On this layer, check:

- Local network configuration of the system under test:
  - IP address and subnet mask
  - Default gateway
  - DNS servers

- Routing within the LAN:
  - Can you ping the default gateway and DNS servers?
  - Can you ping another system on the same network?
  - Does the DNS server respond to queries?

It is important to know the expected values for these settings. For example, you may see that a system does have an IP address, say, 169.254.152.12, but no internet access. If you don't know that this particular address range is used for providing a host with an IP address when no DHCP server is available, you might spend a lot of time troubleshooting the wrong issue.

29.4.1. checking the local network configuration

- IP address and subnet mask: `ip address`
  - Do you have an IP address?
    - If not, the DHCP server may be down, is unreachable from this system, or doesn't give an IP address to this system.
    - The problem may also be on the DHCP server side.
  - Is it in the correct subnet?
    - If not, check if the system has an "link-local" address, e.g. in the range 169.254.0.0/16. This is a sign that the system couldn't get an IP address from a DHCP server.
    - If the address is plain wrong, check the configuration of the network interface. Maybe you configured a static IP address when the system should get one from DHCP, or maybe you forgot to restart the network service after a config change.
  - Is the subnet mask correct? E.g. the IP is correct, but pinging another system in the LAN gives a "network unreachable" error.
    - If not, routing within the LAN may not be possible.
    - Check the configuration of the network interface.
- Default gateway: `ip route`
  - Is the default gateway present?
  - Is it the correct IP address in the expected subnet?
- DNS servers: `cat /etc/resolv.conf` or `resolvectl dns`
  - Is the `nameserver` entry present
  - Is the DNS server IP correct?

The IP addresses on the database server in our lab setup:

```
[vagrant@dbt ~]$ ip -br a
lo      UNKNOWN    127.0.0.1/8 ::1/128
eth0    UP         10.0.2.15/24 fe80::4de4:bee5:560:8219/64
eth1    UP         192.168.76.73/24 fe80::a00:27ff:fe8a:f6f2/64
```

The loopback interface always has the address 127.0.0.1. The `eth0` interface has the address 10.0.2.15/24, which is the default for the NAT interface in VirtualBox. The `eth1` interface has the address 192.168.76.73, which is the one we expect. The subnet mask is /24, the default for a private class C network.

However, if we look at the web server:

```
[vagrant@webt ~]$ ip -br a
lo      UNKNOWN    127.0.0.1/8 ::1/128
eth0    UP         10.0.2.15/24 fe80::399a:5f3c:2fc7:9979/64
eth1    UP         192.168.56.172/25 fe80::a00:27ff:feb3:29e3/64
```

The `eth0` interface also has the address 10.0.2.15/24, but that's not actually a problem. This is the case for *all* VirtualBox VMs attached to a NAT interface. The VM is isolated in this network, but can reach the Internet through the host system. Since it's a separate network, it does not interfere with the address on the other VM.

However, `eth1` is clearly wrong. The address is in the wrong subnet. Let's assign the correct address:

```
[vagrant@webt ~]$ nmcli device status
DEVICE  TYPE      STATE                   CONNECTION
eth0    ethernet  connected               eth0
eth1    ethernet  connected               System eth1
lo      loopback  connected (externally)  lo
[vagrant@webt ~]$ nmcli -f IP4 connection show 'System eth1'
IP4.ADDRESS[1]:    192.168.56.172/25
IP4.GATEWAY:       --
IP4.ROUTE[1]:      dst = 192.168.56.128/25, nh = 0.0.0.0, mt = 101
[vagrant@webt ~]$ sudo nmcli connection modify 'System eth1' ipv4.addresses 192.168.76.72/2
[vagrant@webt ~]$ sudo nmcli connection down 'System eth1'
Connection 'System eth1' successfully deactivated (D-Bus active path: /org/freedesktop/Netw
[vagrant@webt ~]$ sudo nmcli connection up 'System eth1'
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/Acti
[vagrant@webt ~]$ ip -br a
lo      UNKNOWN    127.0.0.1/8 ::1/128
eth0    UP         10.0.2.15/24 fe80::399a:5f3c:2fc7:9979/64
eth1    UP         192.168.76.72/24 fe80::a00:27ff:feb3:29e3/64
```

Remark that this IP address had another important issue, i.e. the network mask was incorrect. In order for two hosts on the same LAN/subnet to be able to communicate, the network part of their IP address must be the same. If the subnet masks do not match, it's possible that another host on the network is not considered to be on the same network, causing the packet to be sent to the default gateway instead of being delivered directly.

Next, let's check the routing table:

```
[vagrant@webt ~]$ ip route
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
192.168.76.0/24 dev eth1 proto kernel scope link src 192.168.76.72 metric 101
```

This routing table is correct for this VM. The VM is attached to two network interfaces, `eth0` and `eth1`. IP packets destined for address 10.0.2.0/24 (the subnet of the NAT interface) are sent out through `eth0`, and packets destined for 192.168.76.0/24 (the subnet of the Host-only Adapter) are sent out through `eth1`. All other packets are sent to the default gateway, 10.0.2.2. This is the default IP address of the gateway simulated by the VirtualBox NAT interface.

Finally, let's check the DNS configuration:

```
[vagrant@webt ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 10.0.2.3
```

Again, this is correct. The DNS server is the one provided by the VirtualBox NAT interface, and has always the IP address 10.0.2.3.

29.4.2. routing within the LAN

- Ping between hosts within the LAN
- Ping the default gateway
  - Remark: some system administrators deliberately disable ICMP echo requests (ping) on network devices for security purposes. If you can't ping the default gateway, it may be that the device is configured this way.
- Ping the DNS server
- Query the DNS server (with `dig` or `nslookup`)

Let's try this on the `webt` VM:

```
[vagrant@webt ~]$ ping -c1 192.168.76.73
PING 192.168.76.73 (192.168.76.73) 56(84) bytes of data.
64 bytes from 192.168.76.73: icmp_seq=1 ttl=64 time=0.974 ms

--- 192.168.76.73 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.974/0.974/0.974/0.000 ms
[vagrant@webt ~]$ ping -c1 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.490 ms

--- 10.0.2.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.490/0.490/0.490/0.000 ms
[vagrant@webt ~]$ ping -c1 10.0.2.3
PING 10.0.2.3 (10.0.2.3) 56(84) bytes of data.
64 bytes from 10.0.2.3: icmp_seq=1 ttl=64 time=0.706 ms

--- 10.0.2.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.706/0.706/0.706/0.000 ms
[vagrant@webt ~]$ dig @10.0.2.3 www.linux-training.be +short
188.40.26.208
```

We receive a ping response from the database server, the default gateway and the DNS server. The DNS server also responds to queries.

29.4.3. internet connectivity

After this step, you might want to test Internet connectivity by e.g. pinging the WAN interface of the default gateway, or the IP address of the next hop (if you know it).

If the previous steps were successful, but you can't reach the Internet, the problem may be with the default gateway or the routing table on the default gateway.

This is outside the scope of this troubleshooting guide, since we're focusing on troubleshooting services we are setting up ourselves.

29.5. the transport layer

On this layer, check:

- Is the service running?
- Is the service listening on the correct port? `sudo ss -tulpn`
- Does the firewall allow traffic to the service port? `sudo firewall-cmd --list-all`

29.5.1. is the service running?

Use the command `systemctl status <service>`. Check for `active (running)` in the output, or `inactive (dead)`. If the service is not running, you can try to start it with `sudo systemctl start <service>`. If this fails, check the logs for error messages.

Also check whether the service is `enabled`, i.e. will start automatically when the system boots. If not, enable it with `sudo systemctl enable <service>`.

Let's check whether MariaDB is running on the database server:

```
[vagrant@dbt ~]$ systemctl status mariadb
● mariadb.service - MariaDB 10.5 database server
    Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-06-17 13:11:15 UTC; 33min ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Process: 730 ExecStartPre=/usr/libexec/mariadb-check-socket (code=exited, status=0/SUCC
   Process: 773 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir mariadb.service (code=exi
   Process: 853 ExecStartPost=/usr/libexec/mariadb-check-upgrade (code=exited, status=0/SU
  Main PID: 814 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 8 (limit: 11128)
     Memory: 103.4M
        CPU: 632ms
     CGroup: /system.slice/mariadb.service
             └─814 /usr/libexec/mariadbd --basedir=/usr
```

MariaDB is running correctly!

For the web server, we get:

```
[vagrant@webt ~]$ systemctl status httpd
○ httpd.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: inactive (dead)
       Docs: man:httpd.service(8)
```

So Apache is not running. We can also see that it is disabled, so it won't start automatically when the system boots. Let's start and enable it:

```
[vagrant@webt ~]$ sudo systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/syste
[vagrant@webt ~]$ systemctl is-active httpd
active
```

Starting Apache succeeded, and it is now running and enabled. This time, we used the `is-active` command to avoid the long output of `systemctl status`. However, you can use `status` to get more information about the service.

29.5.2. is the service listening on the correct port?

Use the command `sudo ss -tulpn` (show sockets). The options have the following meaning:

- `-t`: show TCP sockets.
- `-u`: show UDP sockets (not relevant for pure TCP-based services like HTTP(S), can be omitted in that case).
- `-l`: show only listening (server) sockets.
- `-p`: show the process that owns the socket. This requires root privileges, hence the `sudo` before the command.
- `-n`: show IP addresses instead of trying to resolve them to host names and show port numbers instead of the service name (as enumerated in the file `/etc/services`).

The order of the options is not important, but for Dutch speaking people, the mnemonic "TULPeN" may help to remember the options.

You need to know which interfaces and ports the service is supposed to listen on! If the service is running, but not listening on the expected port (e.g. 8443 instead of 443 for a web server). Some services are configured to only listen on the loopback interface, which also can be determined with `ss`.

We know that MariaDB listens on port 3306, and a web server on port 80 and/or 443. Let's check the database server first. Remark that we don't use `-u` here, since both MariaDB and HTTP(S) are TCP-based services.

```
[vagrant@dbt ~]$ sudo ss -tlnp | grep mariadb
LISTEN 0    80    *:3306    *:*    users:(("mariadbd",pid=814,fd=20))
```

The important part is the `*:3306`, which means that MariaDB is listening on all interfaces (`*`) on port 3306. The process that owns the socket is `mariadbd`.

On the web server, we get (some superfluous output omitted and replaced with `...`):

```
[vagrant@webt ~]$ sudo ss -tlnp | grep httpd
LISTEN 0    511    *:443     *:*    users:(("httpd",pid=7672,fd=6),...)
LISTEN 0    511    *:8080    *:*    users:(("httpd",pid=7672,fd=4),...)
```

The `httpd` proces listens on port 443 and 8080. This is not what we expect. The default ports for HTTP and HTTPS are 80 and 443, respectively. 8080 is a common alternative for test setups, but it's not the default, so won't work if we want to acces the web server with a browser (unless we specify the port in the URL, but that's not what we want here).

Let's check the configuration file of Apache:

```
[vagrant@webt ~]$ grep 8080 /etc/httpd/conf/httpd.conf
Listen 8080
```

Indeed, Apache is configured to listen on port 8080. Edit the configuration file and change the port to 80. After that, restart the service:

```
[vagrant@webt ~]$ sudo vi /etc/httpd/conf/httpd.conf
... (make the necessary changes and save) ...
[vagrant@webt ~]$ apachectl configtest
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
Syntax OK
[vagrant@webt ~]$ sudo systemctl restart httpd
sudo systemctl restart httpd
[vagrant@webt ~]$ sudo ss -tlnp | grep httpd
LISTEN 0    511    *:80     *:*    users:(("httpd",pid=7977,fd=4),...)
LISTEN 0    511    *:443    *:*    users:(("httpd",pid=7977,fd=6),...)
```

Note that we executed `apachectl configtest` before restarting the service. This command checks the syntax of the configuration file. If there are errors, the service won't start. If the syntax is OK, the command will return `Syntax OK`. The warning about the server name is normal for a test environment, so can be safely ignored. Now, Apache listens on the correct ports.

29.5.3. check the firewall configuration

Use the command `sudo firewall-cmd --list-all` to check the firewall configuration. This command shows all zones and the services that are allowed in each zone. If the service is not listed, it is not allowed through the firewall.

On the database server, we get:

```
[vagrant@dbt ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources:
  services: cockpit dhcpv6-client mysql ssh
  ports:
...
```

We see `mysql` in the list of services, which is correct for MariaDB. On the web server, we get:

```
[vagrant@webt ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
...
```

There is no `http` or `https` service in the list. We need to add these services to the firewall configuration and reload the firewall rules:

```
[vagrant@webt ~]$ sudo firewall-cmd --add-service=http --permanent
success
[vagrant@webt ~]$ sudo firewall-cmd --add-service=https --permanent
success
[vagrant@webt ~]$ sudo firewall-cmd --reload
success
[vagrant@webt ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports:
  protocols:
...
```

Now HTTP and HTTPS traffic is allowed through the firewall.

At this time, the web server should be available for users on the local network. You can test this by opening a browser on the host system and navigating to http://192.168.76.72. If you see the Apache test page, everything up to this point is working correctly.

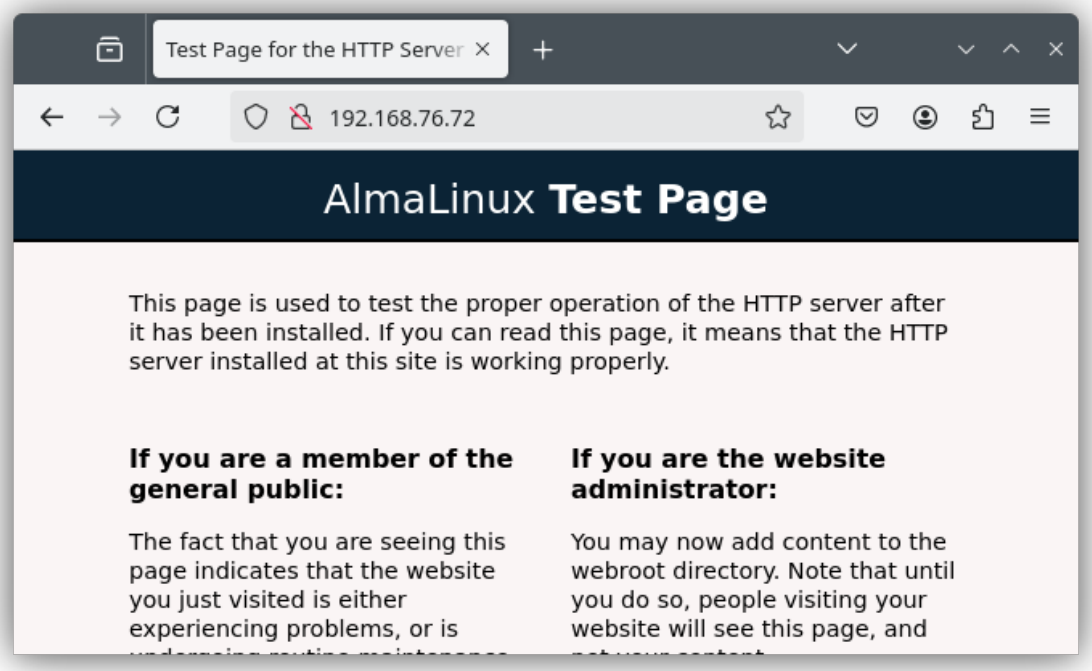


Figure 29.2.: Apache test page

29.6. the application layer

The application layer is the most diverse, as it comprises all services that run on top of the transport layer. This includes web servers, mail servers, DNS servers, etc. Troubleshooting specific behaviour of these services is beyond the scope of this chapter, but there are a few things that you should always check, regardless of the service:

- Check the logs
- Validate the syntax of the configuration files
- Read the manual
- Use command-line client tools to test the service

All other checks are specific to the application.

29.6.1. check the logs

Linux systems based on `systemd` use `journalctl` to manage system logs. You can use `journalctl` (with root privileges) to view the logs of a specific service, e.g. `sudo journalctl -u <service>`. The `-f` option will show new log entries as they are written and the `-l` option will show the full log entries (instead of truncating lines that do not fit within the width of the terminal).

Some services also write logs to a file in `/var/log`. For example, when troubleshooting a web server, you might want to check `/var/log/httpd/error_log` for messages that do not appear with `journalctl`. Check the documentation of the service to find out where the logs are stored.

Following log files in real-time can be done with the command `tail -f /var/log/<logfile>`.

29.6.2. validate config file syntax

Most services have a command that checks the syntax of a configuration file before/without starting the service. It is useful to always run this command first to check for existing errors. Also, after making any change to the configuration file, run the command again to ensure you don't introduce new errors.

A few examples:

- Apache web server: `sudo apachectl configtest`
- Nginx web server: `sudo nginx -t`
- Bind DNS server: `sudo named-checkconf` and `sudo named-checkzone <zone> <zone-file>`
- Vsftpd FTP server: `sudo vsftpd -t`

Check the manual of the service for the exact command to use.

29.6.3. read the manual

When you are responsible for managing a service in production, you must know the system inside out in order to be effective in your job. At a certain point, googling for solutions will no longer be sufficient. You need to know the service's configuration file, log files, and the commands to manage the service.

That's information that you won't find on Stack Overflow or Reddit, but in the manuals. Finding help is a topic that is covered in another chapter, so we won't go into detail here. A few suggestions and examples without striving to be comprehensive:

- Check the man page of the command, configuration file or service
- Check the documentation of your Linux distribution (e.g. the RedHat Manuals or Debian Documentation)
- Check the reference manual of the service (e.g. the Apache HTTP Server Documentation, the Nginx Documentation, the Bind 9 Administrator Reference Manual)
- Read a (good) book about the service, e.g. DNS for Rocket Scientists

29.6.4. use command-line tools

For troubleshooting purposes, it is often more useful to use command-line client tools than the graphical user interface that you would run for daily use. GUIs often hide important details, or give only generic error messages.

The specific tools you can use depend on the service you are troubleshooting and there are a lot. Here are a few examples, again without trying to be complete:

- `curl` for web servers
- `dig` for DNS servers
- `smbclient` for Samba file servers
- `netcat` for general network troubleshooting, setting up raw TCP connections
- `nmap` for network scanning (which also includes scripts to test specific services)

29.7. SELinux troubleshooting

Although the default web page of Apache should be visible now, when we try to open the PHP test script at http://192.168.76.72/test.php, we get an "Access denied" error. Let's investigate the cause and fix it.

TODO

29.8. general guidelines

To conclude, here are some general guidelines that you should use when troubleshooting issues with network services:

- Before making any change, **back up** configuration files.
- Always **be systematic**, follow the bottom-up approach.
- Be **thorough**, don't skip any step. You can't solve problems higher up in the stack if issues lower down are not resolved.
- **Do not assume: test!** The fact that "it doesn't work" means that one of your assumptions is wrong.
- **Know your environment.** What is the topology? What are the IP addresses (summarize them in a table)? What services should be running? Read the documentation of the services you are responsible for.
- **Read The Fine Log Files!** You need to be able to interpret the log files of the services you are troubleshooting. Keep a real time log viewer open in a separate terminal window, study the effect that interventions on a system have in the logs.
- **Read The Fine Error Message!** Error messages are essential to understand what is going wrong, and on which layer in the TCP/IP stack the issue is located.
- Work in **small steps**. Make one change at a time, and immediately test the effect of that change.
- If possible, **validate config file syntax** after each change and before restarting a service.
- Don't forget to **restart the service** after making changes to their configuration files.
- **Verify each change** so you don't introduce new issues.
- Keep a **cheat sheet** or **troubleshooting checklist** with all the steps to follow and commands you need. For an example, see https://github.com/bertvv/cheat-sheets
- **Automate** to avoid human error. Use a configuration management system to describe the desired state of the infrastructure under your control. Automate tests as well, or implement a monitoring and log aggregation solution.
- Finally, **don't ping Google** as your first troubleshooting step. If it doesn't work (and it probably wont, or you wouldn't need to troubleshoot), too many things could have gone wrong. Go back to the beginning of this list and start there... ;-)

29.9. Exercises

TODO

29.10. Solutions

TODO

Part X.

Storage management

30. disk devices

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This chapter teaches you how to locate and recognise `hard disk devices`. This prepares you for the next chapter, where we put `partitions` on these devices.

30.1. terminology

30.1.1. platter, head, track, cylinder, sector

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

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

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

30.1.2. ide or scsi

Actually, the title should be `ata` or `scsi`, since ide is an ata compatible device. Most desktops use `ata devices`, most servers use `scsi`.

30.1.3. ata

An `ata controller` allows two devices per bus, one `master` and one `slave`. Unless your controller and devices support `cable select`, you have to set this manually with jumpers.

With the introduction of `sata` (serial ata), the original ata was renamed to `parallel ata`. Optical drives often use `atapi`, which is an ATA interface using the SCSI communication protocol.

30.1.4. scsi

A `scsi controller` allows more than two devices. When using `SCSI (small computer system interface)`, each device gets a unique `scsi id`. The `scsi controller` also needs a `scsi id`, do not use this id for a scsi-attached device.

Older 8-bit SCSI is now called `narrow`, whereas 16-bit is `wide`. When the bus speeds was doubled to 10Mhz, this was known as `fast SCSI`. Doubling to 20Mhz made it `ultra SCSI`. Take a look at http://en.wikipedia.org/wiki/SCSI for more SCSI standards.

30.1.5. block device

Random access hard disk devices have an abstraction layer called `block device` to enable formatting in fixed-size (usually 512 bytes) blocks. Blocks can be accessed independent of access to other blocks.

```
[root@linux ~]# lsblk
NAME                          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                             8:0    0   40G  0 disk
--sda1                          8:1    0  500M  0 part /boot
--sda2                          8:2    0 39.5G  0 part
  --VolGroup-lv_root (dm-0) 253:0    0 38.6G  0 lvm  /
  --VolGroup-lv_swap (dm-1) 253:1    0  928M  0 lvm  [SWAP]
sdb                             8:16   0   72G  0 disk
sdc                             8:32   0  144G  0 disk
```

A block device has the letter b to denote the file type in the output of `ls -l`.

```
[root@linux ~]# ls -l /dev/sd*
brw-rw----. 1 root disk 8,  0 Apr 19 10:12 /dev/sda
brw-rw----. 1 root disk 8,  1 Apr 19 10:12 /dev/sda1
brw-rw----. 1 root disk 8,  2 Apr 19 10:12 /dev/sda2
brw-rw----. 1 root disk 8, 16 Apr 19 10:12 /dev/sdb
brw-rw----. 1 root disk 8, 32 Apr 19 10:12 /dev/sdc
```

Virtual devices like `raid` or `lvm` are also listed as `block devices` as seen in this RHEL7 virtual machine.

```
[root@linux ~]# lsblk
NAME                    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                       8:0    0     8G  0 disk
├─sda1                    8:1    0   400M  0 part
│ └─md0                   9:0    0 399.7M  0 raid1
├─sda2                    8:2    0   400M  0 part
│ └─md0                   9:0    0 399.7M  0 raid1
└─sda3                    8:3    0   400M  0 part
sdb                       8:16   0     8G  0 disk
sdc                       8:32   0     8G  0 disk
sdd                       8:48   0     2G  0 disk
sde                       8:64   0     2G  0 disk
sdf                       8:80   0  20.5G  0 disk
├─sdf1                    8:81   0   500M  0 part  /boot
└─sdf2                    8:82   0    20G  0 part
  ├─centos_centos7-swap 253:0    0     2G  0 lvm   [SWAP]
  └─centos_centos7-root 253:1    0    18G  0 lvm   /
sr0                      11:0    1  1024M  0 rom
[root@linux ~]#
```

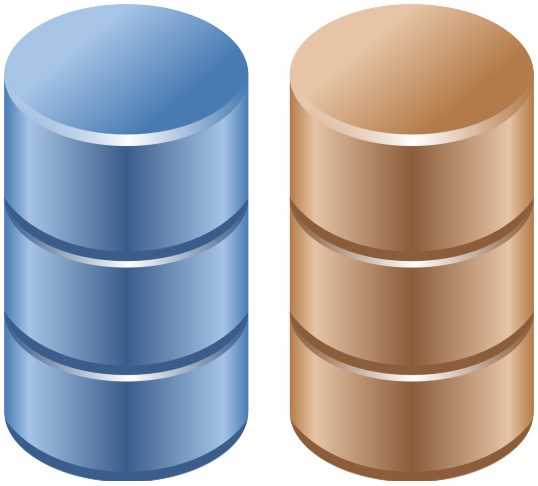
Note that a `character device` is a constant stream of characters, being denoted by a c in `ls -l`. Note also that the `ISO 9660` standard for cdrom uses a `2048 byte` block size.

Old hard disks (and floppy disks) use `cylinder-head-sector` addressing to access a sector on the disk. Most current disks use `LBA (Logical Block Addressing)`.

30.1.6. solid state drive

A `solid state drive` or `ssd` is a block device without moving parts. It is comparable to `flash memory`. An `ssd` is more expensive than a hard disk, but it typically has a much faster access time.

In this book we will use the following pictograms for `spindle disks` (in brown) and `solid state disks` (in blue).



30.2. device naming

30.2.1. ata (ide) device naming

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

Table 30.1.: ide device naming

controller	connection	device name
ide0	master	/dev/hda
	slave	/dev/hdb
ide1	master	/dev/hdc
	slave	/dev/hdd

It is possible to have only `/dev/hda` and `/dev/hdd`. The first one is a single ata hard disk, the second one is the cdrom (by default configured as slave).

30.2.2. scsi device naming

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

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

Table 30.2.: scsi device naming

device	scsi id	device name
disk 0	0	/dev/sda
disk 1	1	/dev/sdb
raid controller 0	5	/dev/sdc
raid controller 1	6	/dev/sdd

A modern Linux system will use `/dev/sd*` for scsi and sata devices, and also for sd-cards, usb-sticks, (legacy) ATA/IDE devices and solid state drives.

30.3. discovering disk devices

30.3.1. fdisk

You can start by using `/sbin/fdisk` to find out what kind of disks are seen by the kernel. Below the result on old Debian desktop, with two `ata-ide disks` present.

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

And here an example of `sata and scsi disks` on a server with CentOS. Remember that `sata` disks are also presented to you with the `scsi` /dev/sd* notation.

```
[root@linux ~]# fdisk -l | grep 'Disk /dev/sd'
Disk /dev/sda: 42.9 GB, 42949672960 bytes
Disk /dev/sdb: 77.3 GB, 77309411328 bytes
Disk /dev/sdc: 154.6 GB, 154618822656 bytes
Disk /dev/sdd: 154.6 GB, 154618822656 bytes
```

Here is an overview of disks on a RHEL4u3 server with two real 72GB `scsi disks`. This server is attached to a `NAS` with four `NAS disks` of half a terabyte. On the NAS disks, four LVM (/dev/mdx) software RAID devices are configured.

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

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

```
[root@linux ~]# fdisk -l /dev/sdc

Disk /dev/sdc: 154.6 GB, 154618822656 bytes
255 heads, 63 sectors/track, 18798 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0×00000000
```

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

30.3.2. dmesg

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

```
[root@linux ~]# dmesg | grep 'sd[a-z]' | head
sd 0:0:0:0: [sda] 83886080 512-byte logical blocks: (42.9 GB/40.0 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support \
DPO or FUA
sda: sda1 sda2
sd 0:0:0:0: [sda] Attached SCSI disk
sd 3:0:0:0: [sdb] 150994944 512-byte logical blocks: (77.3 GB/72.0 GiB)
sd 3:0:0:0: [sdb] Write Protect is off
sd 3:0:0:0: [sdb] Mode Sense: 00 3a 00 00
sd 3:0:0:0: [sdb] Write cache: enabled, read cache: enabled, doesn't support \
DPO or FUA
```

Here is another example of `dmesg` on a computer with a 200GB ata disk.

```
student@linux:~$ dmesg | grep -i "ata disk"
[    2.624149] hda: ST360021A, ATA DISK drive
[    2.904150] hdb: Maxtor 6Y080L0, ATA DISK drive
[    3.472148] hdd: WDC WD2000BB-98DWA0, ATA DISK drive
```

Third and last example of `dmesg` running on RHEL5.3.

```
root@linux ~# dmesg | grep -i "scsi disk"
sd 0:0:2:0: Attached scsi disk sda
sd 0:0:3:0: Attached scsi disk sdb
sd 0:0:6:0: Attached scsi disk sdc
```

30.3.3. /sbin/lshw

The `lshw` tool will `list hardware`. With the right options `lshw` can show a lot of information about disks (and partitions).

Below a truncated screenshot on Debian 6:

```
root@linux~# lshw -class volume | grep -A1 -B2 scsi
       description: Linux raid autodetect partition
       physical id: 1
       bus info: scsi@1:0.0.0,1
       logical name: /dev/sdb1
--
       description: Linux raid autodetect partition
       physical id: 1
       bus info: scsi@2:0.0.0,1
       logical name: /dev/sdc1
--
       description: Linux raid autodetect partition
       physical id: 1
       bus info: scsi@3:0.0.0,1
       logical name: /dev/sdd1
--
       description: Linux raid autodetect partition
       physical id: 1
       bus info: scsi@4:0.0.0,1
       logical name: /dev/sde1
--
       vendor: Linux
       physical id: 1
       bus info: scsi@0:0.0.0,1
       logical name: /dev/sda1
--
       vendor: Linux
       physical id: 2
       bus info: scsi@0:0.0.0,2
       logical name: /dev/sda2
--
       description: Extended partition
       physical id: 3
       bus info: scsi@0:0.0.0,3
       logical name: /dev/sda3
```

Redhat and CentOS do not have this tool (unless you add a repository).

30.3.4. /sbin/lsscsi

The `lsscsi` command provides a nice readable output of all scsi (and scsi emulated devices). This first screenshot shows `lsscsi` on a SPARC system.

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

Below a screenshot of `lsscsi` on a QNAP NAS (which has four 750GB disks and boots from a usb stick).

```
lroot@linux~# lsscsi
[0:0:0:0]    disk    SanDisk  Cruzer Edge      1.19  /dev/sda
[1:0:0:0]    disk    ATA      ST3750330AS      SD04  /dev/sdb
[2:0:0:0]    disk    ATA      ST3750330AS      SD04  /dev/sdc
[3:0:0:0]    disk    ATA      ST3750330AS      SD04  /dev/sdd
[4:0:0:0]    disk    ATA      ST3750330AS      SD04  /dev/sde
```

This screenshot shows the classic output of `lsscsi`.

```
root@linux~# lsscsi -c
Attached devices:
Host: scsi0 Channel: 00 Target: 00 Lun: 00
  Vendor: SanDisk  Model: Cruzer Edge      Rev: 1.19
  Type:   Direct-Access                    ANSI SCSI revision: 02
Host: scsi1 Channel: 00 Target: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi2 Channel: 00 Target: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi3 Channel: 00 Target: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi4 Channel: 00 Target: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI SCSI revision: 05
```

30.3.5. /proc/scsi/scsi

Another way to locate `scsi` (or sd) devices is via `/proc/scsi/scsi`.

This screenshot is from a `sparc` computer with adaptec RAID5.

```
root@shaka:~# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: Adaptec  Model: RAID5           Rev: V1.0
  Type:   Direct-Access                   ANSI SCSI revision: 02
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: SEAGATE  Model: ST336605FSUN36G  Rev: 0438
  Type:   Direct-Access                   ANSI SCSI revision: 03
root@shaka:~#
```

Here we run `cat /proc/scsi/scsi` on the QNAP from above (with Debian Linux).

```
root@linux~# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: SanDisk  Model: Cruzer Edge      Rev: 1.19
  Type:   Direct-Access                    ANSI  SCSI revision: 02
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi2 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi3 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi4 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3750330AS      Rev: SD04
  Type:   Direct-Access                    ANSI  SCSI revision: 05
```

Note that some recent versions of Debian have this disabled in the kernel. You can enable it (after a kernel compile) using this entry:

```
# CONFIG_SCSI_PROC_FS is not set
```

Redhat and CentOS have this by default (if there are scsi devices present).

```
[root@linux ~]# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: VBOX HARDDISK    Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi3 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: VBOX HARDDISK    Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi4 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: VBOX HARDDISK    Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
```

30.4. erasing a hard disk

Before selling your old hard disk on the internet, it may be a good idea to erase it. By simply repartitioning, or by using the Microsoft Windows format utility, or even after an `mkfs` command, some people will still be able to read most of the data on the disk.

```
root@linux~# aptitude search foremost autopsy sleuthkit | tr -s ' '
p autopsy - graphical interface to SleuthKit
p foremost - Forensics application to recover data
p sleuthkit - collection of tools for forensics analysis
```

Although technically the `/sbin/badblocks` tool is meant to look for bad blocks, you can use it to completely erase all data from a disk. Since this is really writing to every sector of the disk, it can take a long time!

```
root@linux:~# badblocks -ws /dev/sdb
Testing with pattern 0×aa: done
Reading and comparing: done
Testing with pattern 0×55: done
Reading and comparing: done
Testing with pattern 0×ff: done
Reading and comparing: done
Testing with pattern 0×00: done
Reading and comparing: done
```

The previous screenshot overwrites every sector of the disk `four times`. Erasing `once` with a tool like `dd` is enough to destroy all data.

Warning, this screenshot shows how to permanently destroy all data on a block device.

```
[root@linux ~]# dd if=/dev/zero of=/dev/sdb
```

30.5. advanced hard disk settings

Tweaking of hard disk settings (dma, gap, ...) are not covered in this course. Several tools exists, `hdparm` and `sdparm` are two of them.

`hdparm` can be used to display or set information and parameters about an ATA (or SATA) hard disk device. The -i and -I options will give you even more information about the physical properties of the device.

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

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

Below `hdparm` info about a 200GB IDE disk.

```
root@linux:~# hdparm /dev/hdd

/dev/hdd:
 multcount     =  0 (off)
 IO_support    =  0 (default)
 unmaskirq     =  0 (off)
 using_dma     =  1 (on)
 keepsettings  =  0 (off)
 readonly      =  0 (off)
 readahead     = 256 (on)
 geometry      = 24321/255/63, sectors = 390721968, start = 0
```

Here a screenshot of `sdparm` on Ubuntu 10.10.

```
root@linux:~# aptitude install sdparm
...
root@linux:~# sdparm /dev/sda | head -1
    /dev/sda: ATA       FUJITSU MJA2160B  0081
root@linux:~# man sdparm
```

Use `hdparm` and `sdparm` with care.

30.6. practice: hard disk devices

`About this lab:` To practice working with hard disks, you will need some hard disks. When there are no physical hard disk available, you can use virtual disks in `vmware` or `VirtualBox`. The teacher will help you in attaching a couple of ATA and/or SCSI disks to a virtual machine. The results of this lab can be used in the next three labs (partitions, file systems, mounting).

It is adviced to attach three 1GB disks and three 2GB disks to the virtual machine. This will allow for some freedom in the practices of this chapter as well as the next chapters (raid, lvm, iSCSI).

1. Use `dmesg` to make a list of hard disk devices detected at boot-up.

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

3. Stop a virtual machine, add three virtual 1 gigabyte `scsi` hard disk devices and one virtual 400 megabyte `ide` hard disk device. If possible, also add another virtual 400 megabyte `ide` disk.

4. Use `dmesg` to verify that all the new disks are properly detected at boot-up.

5. Verify that you can see the disk devices in `/dev`.

6. Use `fdisk` (with `grep` and `/dev/null`) to display the total size of the new disks.

7. Use `badblocks` to completely erase one of the smaller hard disks.

8. Look at `/proc/scsi/scsi`.

9. If possible, install `lsscsi`, `lshw` and use them to list the disks.

30.7. solution: hard disk devices

1. Use `dmesg` to make a list of hard disk devices detected at boot-up.

```
Some possible answers ...

dmesg | grep -i disk

Looking for ATA disks: dmesg | grep hd[abcd]

Looking for ATA disks: dmesg | grep -i "ata disk"

Looking for SCSI disks: dmesg | grep sd[a-f]

Looking for SCSI disks: dmesg | grep -i "scsi disk"
```

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

```
fdisk -l
```

3. Stop a virtual machine, add three virtual 1 gigabyte `scsi` hard disk devices and one virtual 400 megabyte `ide` hard disk device. If possible, also add another virtual 400 megabyte `ide` disk.

```
This exercise happens in the settings of vmware or VirtualBox.
```

4. Use `dmesg` to verify that all the new disks are properly detected at boot-up.

```
See 1.
```

5. Verify that you can see the disk devices in `/dev`.

```
SCSI+SATA: ls -l /dev/sd*

ATA: ls -l /dev/hd*
```

6. Use `fdisk` (with `grep` and `/dev/null`) to display the total size of the new disks.

```
root@linux ~# fdisk -l 2>/dev/null | grep [MGT]B
Disk /dev/hda: 21.4 GB, 21474836480 bytes
Disk /dev/hdb: 1073 MB, 1073741824 bytes
Disk /dev/sda: 2147 MB, 2147483648 bytes
Disk /dev/sdb: 2147 MB, 2147483648 bytes
Disk /dev/sdc: 2147 MB, 2147483648 bytes
```

7. Use `badblocks` to completely erase one of the smaller hard disks.

```
#Verify the device (/dev/sdc??) you want to erase before typing this.
#
root@linux ~# badblocks -ws /dev/sdc
Testing with pattern 0×aa: done
Reading and comparing: done
Testing with pattern 0×55: done
Reading and comparing: done
Testing with pattern 0×ff: done
Reading and comparing: done
Testing with pattern 0×00: done
Reading and comparing: done
```

8. Look at `/proc/scsi/scsi`.

```
root@linux ~# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 02 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi0 Channel: 00 Id: 03 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi0 Channel: 00 Id: 06 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI SCSI revision: 05
```

9. If possible, install `lsscsi`, `lshw` and use them to list the disks.

```
Debian,Ubuntu: aptitude install lsscsi lshw

Fedora: yum install lsscsi lshw

root@linux ~# lsscsi
[0:0:2:0]    disk    VBOX     HARDDISK         1.0   /dev/sda
[0:0:3:0]    disk    VBOX     HARDDISK         1.0   /dev/sdb
[0:0:6:0]    disk    VBOX     HARDDISK         1.0   /dev/sdc
```

31. disk partitions

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This chapter continues on the `hard disk devices` from the previous one. Here we will put `partitions` on those devices.

This chapter prepares you for the next chapter, where we put `file systems` on our partitions.

31.1. about partitions

31.1.1. primary, extended and logical

Linux requires you to create one or more `partitions`. The next paragraphs will explain how to create and use partitions.

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

Table 31.1.: primary, extended and logical partitions

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

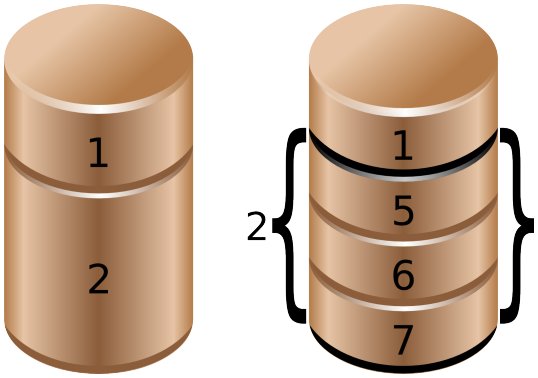
31.1.2. partition naming

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

Table 31.2.: Partition naming

partition	device
/dev/hda1	first primary partition on /dev/hda
/dev/hda2	second primary or extended partition on /dev/hda
/dev/sda5	first logical drive on /dev/sda
/dev/sdb6	second logical on /dev/sdb

The picture below shows two (spindle) disks with partitions. Note that an extended partition is a container holding logical drives.



31.2. discovering partitions

31.2.1. fdisk -l

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

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

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

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

31.2.2. /proc/partitions

The `/proc/partitions` file contains a table with major and minor number of partitioned devices, their number of blocks and the device name in `/dev`. Verify with `/proc/devices` to link the major number to the proper device.

```
student@linux:~$ cat /proc/partitions
major minor  #blocks  name

   3     0     524288 hda
   3    64     734003 hdb
   8     0    8388608 sda
   8     1     104391 sda1
   8     2    8281507 sda2
   8    16    1048576 sdb
   8    32    1048576 sdc
   8    48    1048576 sdd
 253     0    7176192 dm-0
 253     1    1048576 dm-1
```

The major number corresponds to the device type (or driver) and can be found in /proc/devices. In this case 3 corresponds to ide and 8 to sd. The major number determines the device driver to be used with this device.

The minor number is a unique identification of an instance of this device type. The devices.txt file in the kernel tree contains a full list of major and minor numbers.

31.2.3. parted and others

You may be interested in alternatives to fdisk like parted, cfdisk, sfdisk and gparted. This course mainly uses fdisk to partition hard disks.

parted is recommended by some Linux distributions for handling storage with gpt instead of mbr.

Below a screenshot of parted on CentOS.

```
[root@linux ~]# rpm -q parted
parted-2.1-21.el6.x86_64
[root@linux ~]# parted /dev/sda
GNU Parted 2.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  525MB   524MB   primary  ext4         boot
 2      525MB   42.9GB  42.4GB  primary               lvm

(parted)
```

31.3. partitioning new disks

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

31.3.1. recognising the disk

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

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

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

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

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

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

31.3.2. opening the disk with fdisk

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

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

Warning: invalid flag 0×0000 of partition table 4 will be corrected ...
```

31.3.3. empty partition table

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

```
Command (m for help): p

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

Device Boot      Start         End      Blocks   Id  System
```

31.3.4. create a new partition

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

```
Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-130, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-130, default 130): 14
```

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

```
Command (m for help): p

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

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

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

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

31.3.5. display the new partition

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

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

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

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

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

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

31.4. about the partition table

31.4.1. master boot record

The `partition table` information (primary and extended partitions) is written in the `master boot record` or `mbr`. You can use `dd` to copy the mbr to a file.

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

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

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

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

Or to wipe out the whole partition or disk.

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

31.4.2. partprobe

Don't forget that after restoring a `master boot record` with `dd`, that you need to force the kernel to reread the partition table with `partprobe`. After running `partprobe`, the partitions can be used again.

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

31.4.3. logical drives

The `partition table` does not contain information about `logical drives`. So the `dd` backup of the `mbr` only works for primary and extended partitions. To backup the partition table including the logical drives, you can use `sfdisk`.

This example shows how to backup all partition and logical drive information to a file.

```
sfdisk -d /dev/sda > parttable.sda.sfdisk
```

The following example copies the `mbr` and all `logical drive` info from /dev/sda to /dev/sdb.

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

31.5. GUID partition table

`gpt` was developed because of the limitations of the 1980s `mbr` partitioning scheme (for example only four partitions can be defined, and they have a maximum size two terabytes).

Since 2010 `gpt` is a part of the `uefi` specification, but it is also used on `bios` systems.

Newer versions of `fdisk` work fine with `gpt`, but most production servers today (mid 2015) still have an older `fdisk.`. You can use `parted` instead.

31.6. labeling with parted

`parted` is an interactive tool, just like `fdisk`. Type `help` in `parted` for a list of commands and options.

This screenshot shows how to start `parted` to manage partitions on `/dev/sdb`.

```
[root@linux ~]# parted /dev/sdb
GNU Parted 3.1
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Each command also has built-in help. For example `help mklabel` will list all supported labels. Note that we only discussed `mbr`(msdos) and `gpt` in this book.

```
(parted) help mklabel
 mklabel,mktable LABEL-TYPE          create a new disklabel (partition table)

   LABEL-TYPE is one of: aix, amiga, bsd, dvh, gpt, mac, msdos, pc98, sun, loop
(parted)
```

We create an `mbr` label.

```
(parted) mklabel msdos>
Warning: The existing disk label on /dev/sdb will be destroyed and all data on
this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) mklabel gpt
Warning: The existing disk label on /dev/sdb will be destroyed and all data on
this disk will be lost. Do you want to continue?
Yes/No? Y
(parted)
```

31.6.1. partitioning with parted

Once labeled it is easy to create partitions with `parted`. This screenshot starts with an unpartitioned (but `gpt` labeled) disk.

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start  End  Size  File system  Name  Flags

(parted)
```

This example shows how to create two primary partitions of equal size.

```
(parted) mkpart primary 0 50%
Warning: The resulting partition is not properly aligned for best performance.
Ignore/Cancel? I
(parted) mkpart primary 50% 100%
(parted)
```

Verify with `print` and exit with `quit`. Since `parted` works directly on the disk, there is no need to `w(rite)` like in `fdisk`.

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      17.4kB  4295MB  4295MB               primary
 2      4295MB  8589MB  4294MB               primary

(parted) quit
Information: You may need to update /etc/fstab.

[root@linux ~]#
```

31.7. practice: partitions

1. Use `fdisk -l` to display existing partitions and sizes.
2. Use `df -h` to display existing partitions and sizes.
3. Compare the output of `fdisk` and `df`.
4. Create a 200MB primary partition on a small disk.
5. Create a 400MB primary partition and two 300MB logical drives on a big disk.
6. Use `df -h` and `fdisk -l` to verify your work.
7. Compare the output again of `fdisk` and `df`. Do both commands display the new partitions ?
8. Create a backup with `dd` of the `mbr` that contains your 200MB primary partition.
9. Take a backup of the `partition table` containing your 400MB primary and 300MB logical drives. Make sure the logical drives are in the backup.
10. (optional) Remove all your partitions with fdisk. Then restore your backups.

31.8. solution: partitions

1. Use `fdisk -l` to display existing partitions and sizes.

```
as root: # fdisk -l
```

2. Use `df -h` to display existing partitions and sizes.

```
df -h
```

3. Compare the output of `fdisk` and `df`.

```
Some partitions will be listed in both outputs (maybe /dev/sda1 or /dev/hda1).
```

4. Create a 200MB primary partition on a small disk.

```
Choose one of the disks you added (this example uses /dev/sdc).
root@linux ~# fdisk /dev/sdc
...
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-261, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-261, default 261): +200m
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

5. Create a 400MB primary partition and two 300MB logical drives on a big disk.

```
Choose one of the disks you added (this example uses /dev/sdb)

fdisk /dev/sdb

inside fdisk : n p 1 +400m enter --- n e 2 enter enter --- n l +300m (twice)
```

6. Use `df -h` and `fdisk -l` to verify your work.

```
fdisk -l ; df -h
```

7. Compare the output again of `fdisk` and `df`. Do both commands display the new partitions ?

```
The newly created partitions are visible with fdisk.

But they are not displayed by df.
```

8. Create a backup with `dd` of the `mbr` that contains your 200MB primary partition.

```
dd if=/dev/sdc of=bootsector.sdc.dd count=1 bs=512
```

9. Take a backup of the `partition table` containing your 400MB primary and 300MB logical drives. Make sure the logical drives are in the backup.

```
sfdisk -d /dev/sdb > parttable.sdb.sfdisk
```

32. file systems

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

When you are finished partitioning the hard disk, you can put a `file system` on each partition.

This chapter builds on the `partitions` from the previous chapter, and prepares you for the next one where we will `mount` the filesystems.

32.1. about file systems

A file system is a way of organizing files on your partition. Besides file-based storage, file systems usually include `directories` and `access control`, and contain meta information about files like access times, modification times and file ownership.

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

32.1.1. man fs

The manual page about filesystems is accessed by typing `man fs`.

```
[root@linux ~]# man fs
```

32.1.2. /proc/filesystems

The Linux kernel will inform you about currently loaded file system drivers in `/proc/filesystems`.

```
root@linux ~# cat /proc/filesystems  | grep -v nodev
    ext2
    iso9660
    ext3
```

32.1.3. /etc/filesystems

The `/etc/filesystems` file contains a list of autodetected filesystems (in case the `mount` command is used without the `-t` option.

Help for this file is provided by `man mount`.

```
[root@linux ~]# man mount
```

32.2. common file systems

32.2.1. ext2 and ext3

Once the most common Linux file systems is the `ext2` (the second extended) file system. A disadvantage is that file system checks on ext2 can take a long time.

`ext2` was being replaced by `ext3` on most Linux machines. They are essentially the same, except for the `journaling` which is only present in ext3.

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

32.2.2. creating ext2 and ext3

You can create these file systems with the `/sbin/mkfs` or `/sbin/mke2fs` commands. Use `mke2fs -j` to create an `ext3` file system.

You can convert an ext2 to ext3 with `tune2fs -j`. You can mount an ext3 file system as ext2, but then you lose the journaling. Do not forget to run `mkinitrd` if you are booting from this device.

32.2.3. ext4

The newest incarnation of the ext file system is named `ext4` and is available in the Linux kernel since 2008. `ext4` supports larger files (up to 16 terabyte) and larger file systems than `ext3` (and many more features).

Development started by making `ext3` fully capable for 64-bit. When it turned out the changes were significant, the developers decided to name it `ext4`.

32.2.4. xfs

Redhat Enterprise Linux 7 will have `XFS` as the default file system. This is a highly scalable high-performance file system.

`xfs` was created for `Irix` and for a couple of years it was also used in `FreeBSD`. It is supported by the Linux kernel, but rarely used in dsitributions outside of the Redhat/CentOS realm.

32.2.5. vfat

The `vfat` file system exists in a couple of forms : `fat12` for floppy disks, `fat16` on `ms-dos`, and `fat32` for larger disks. The Linux `vfat` implementation supports all of these, but vfat lacks a lot of features like security and links. `fat` disks can be read by every operating system, and are used a lot for digital cameras, `usb` sticks and to exchange data between different OS'ses on a home user's computer.

32.2.6. iso 9660

`iso 9660` is the standard format for cdroms. Chances are you will encounter this file system also on your hard disk in the form of images of cdroms (often with the .iso extension). The `iso 9660` standard limits filenames to the 8.3 format. The Unix world didn't like this, and thus added the `rock ridge` extensions, which allows for filenames up to 255 characters and Unix-style file-modes, ownership and symbolic links. Another extensions to `iso 9660` is `joliet`, which adds 64 unicode characters to the filename. The `el torito` standard extends `iso 9660` to be able to boot from CD-ROM's.

32.2.7. udf

Most optical media today (including cd's and dvd's) use `udf`, the Universal Disk Format.

32.2.8. swap

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

32.2.9. gfs

Linux clusters often use a dedicated cluster filesystem like GFS, GFS2, ClusterFS, ...

32.2.10. and more...

You may encounter `reiserfs` on older Linux systems. Maybe you will see Sun's `zfs` or the open source `btrfs`. This last one requires a chapter on itself.

32.2.11. /proc/filesystems

The `/proc/filesystems` file displays a list of supported file systems. When you mount a file system without explicitly defining one, then mount will first try to probe `/etc/filesystems` and then probe `/proc/filesystems` for all the filesystems without the `nodev` label. If `/etc/filesystems` ends with a line containing only an asterisk (*) then both files are probed.

```
student@linux:~$ cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   sockfs
nodev   binfmt_misc
nodev   usbfs
nodev   usbdevfs
nodev   futexfs
nodev   tmpfs
nodev   pipefs
nodev   eventpollfs
nodev   devpts
        ext2
nodev   ramfs
```

```
nodev   hugetlbfs
        iso9660
nodev   relayfs
nodev   mqueue
nodev   selinuxfs
        ext3
nodev   rpc_pipefs
nodev   vmware-hgfs
nodev   autofs
student@linux:~$
```

32.3. putting a file system on a partition

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

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

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

```
root@linux:~# mke2fs /dev/sdb1
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
28112 inodes, 112420 blocks
5621 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
14 block groups
8192 blocks per group, 8192 fragments per group
2008 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

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

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

32.4. tuning a file system

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

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

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

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

32.5. checking a file system

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

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

The last column in `/etc/fstab` is used to determine whether a file system should be checked at boot-up.

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

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

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

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

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

check aborted.
```

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

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

32.6. practice: file systems

1. List the filesystems that are known by your system.
2. Create an `ext2` filesystem on the 200MB partition.
3. Create an `ext3` filesystem on one of the 300MB logical drives.
4. Create an `ext4` on the 400MB partition.
5. Set the reserved space for root on the ext3 filesystem to 0 percent.
6. Verify your work with `fdisk` and `df`.
7. Perform a file system check on all the new file systems.

32.7. solution: file systems

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

```
man fs
```

```
cat /proc/filesystems
```

```
cat /etc/filesystems (not on all Linux distributions)
```

2. Create an `ext2` filesystem on the 200MB partition.

```
mke2fs /dev/sdc1 (replace sdc1 with the correct partition)
```

3. Create an `ext3` filesystem on one of the 300MB logical drives.

```
mke2fs -j /dev/sdb5 (replace sdb5 with the correct partition)
```

4. Create an `ext4` on the 400MB partition.

```
mkfs.ext4 /dev/sdb1 (replace sdb1 with the correct partition)
```

5. Set the reserved space for root on the ext3 filesystem to 0 percent.

```
tune2fs -m 0 /dev/sdb5
```

6. Verify your work with `fdisk` and `df`.

```
mkfs (mke2fs) makes no difference in the output of these commands

The big change is in the next topic: mounting
```

7. Perform a file system check on all the new file systems.

```
fsck /dev/sdb1
fsck /dev/sdc1
fsck /dev/sdb5
```

33. mounting

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

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

On all `Unix` systems, every file and every directory is part of one big file tree. To access a file, you need to know the full path starting from the root directory. When adding a `file system` to your computer, you need to make it available somewhere in the file tree. The directory where you make a file system available is called a `mount point`.

33.1. mounting local file systems

33.1.1. mkdir

This example shows how to create a new `mount point` with `mkdir`.

```
root@linux:~# mkdir /home/project42
```

33.1.2. mount

When the `mount point` is created, and a `file system` is present on the partition, then `mount` can `mount` the `file system` on the `mount point directory`.

```
root@linux:~# mount -t ext2 /dev/sdb1 /home/project42/
```

Once mounted, the new file system is accessible to users.

33.1.3. /etc/filesystems

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

When mounting a file system without specifying explicitly the file system, then `mount` will first probe `/etc/filesystems`. Mount will skip lines with the `nodev` directive.

```
student@linux:~$ cat /etc/filesystems
ext3
ext2
nodev proc
nodev devpts
iso9660
vfat
hfs
```

33.1.4. /proc/filesystems

When `/etc/filesystems` does not exist, or ends with a single * on the last line, then `mount` will read `/proc/filesystems`.

```
[root@linux ~]# cat /proc/filesystems | grep -v ^nodev
    ext2
    iso9660
    ext3
```

33.1.5. umount

You can `unmount` a mounted file system using the `umount` command.

```
root@linux:~# umount /home/reet
```

33.2. displaying mounted file systems

To display all mounted file systems, issue the `mount` command. Or look at the files `/proc/mounts` and `/etc/mtab`.

33.2.1. mount

The simplest and most common way to view all mounts is by issuing the `mount` command without any arguments.

```
root@linux:~# mount | grep /dev/sdb
/dev/sdb1 on /home/project42 type ext2 (rw)
```

33.2.2. /proc/mounts

The kernel provides the info in `/proc/mounts` in file form, but `/proc/mounts` does not exist as a file on any hard disk. Looking at `/proc/mounts` is looking at information that comes directly from the kernel.

```
root@linux:~# cat /proc/mounts | grep /dev/sdb
/dev/sdb1 /home/project42 ext2 rw 0 0
```

33.2.3. /etc/mtab

The `/etc/mtab` file is not updated by the kernel, but is maintained by the `mount` command. Do not edit `/etc/mtab` manually.

```
root@linux:~# cat /etc/mtab | grep /dev/sdb
/dev/sdb1 /home/project42 ext2 rw 0 0
```

33.2.4. df

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

```
root@linux:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
11707972   6366996   4746240  58% /
/dev/sda1              101086    9300    86567  10% /boot
none                   127988       0   127988   0% /dev/shm
/dev/sdb1              108865    1550   101694   2% /home/project42
root@linux:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
12G  6.1G  4.6G  58% /
/dev/sda1              99M  9.1M   85M  10% /boot
none                  125M     0  125M   0% /dev/shm
/dev/sdb1             107M  1.6M  100M   2% /home/project42
```

33.2.5. df -h

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

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

33.2.6. du

The `du` command can summarize `disk usage` for files and directories. By using `du` on a mount point you effectively get the disk space used on a file system.

While `du` can go display each subdirectory recursively, the `-s` option will give you a total summary for the parent directory. This option is often used together with `-h`. This means `du -sh` on a mount point gives the total amount used by the file system in that partition.

```
root@linux~# du -sh /boot /srv/wolf
6.2M    /boot
1.1T    /srv/wolf
```

33.3. from start to finish

Below is a screenshot that show a summary roadmap starting with detection of the hardware (/dev/sdb) up until mounting on `/mnt`.

```
[root@linux ~]# dmesg | grep '\[sdb\]'
sd 3:0:0:0: [sdb] 150994944 512-byte logical blocks: (77.3 GB/72.0 GiB)
sd 3:0:0:0: [sdb] Write Protect is off
sd 3:0:0:0: [sdb] Mode Sense: 00 3a 00 00
sd 3:0:0:0: [sdb] Write cache: enabled, read cache: enabled, doesn't support \
```

```
DPO or FUA
sd 3:0:0:0: [sdb] Attached SCSI disk

[root@linux ~]# parted /dev/sdb

(parted) mklabel msdos
(parted) mkpart primary ext4 1 77000
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 77.3GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  77.0GB  77.0GB  primary

(parted) quit
[root@linux ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
4702208 inodes, 18798592 blocks
939929 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
574 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
( output truncated )
...
[root@linux ~]# mount /dev/sdb1 /mnt
[root@linux ~]# mount | grep mnt
/dev/sdb1 on /mnt type ext4 (rw)
[root@linux ~]# df -h | grep mnt
/dev/sdb1              71G  180M   67G   1% /mnt
[root@linux ~]# du -sh /mnt
20K     /mnt
[root@linux ~]# umount /mnt
```

33.4. permanent mounts

Until now, we performed all mounts manually. This works nice, until the next reboot. Luckily there is a way to tell your computer to automatically mount certain file systems during boot.

33.4.1. /etc/fstab

The file system table located in `/etc/fstab` contains a list of file systems, with an option to automtically mount each of them at boot time.

Below is a sample `/etc/fstab` file.

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

By adding the following line, we can automate the mounting of a file system.

```
/dev/sdb1               /home/project42      ext2    defaults    0 0
```

33.4.2. mount /mountpoint

Adding an entry to `/etc/fstab` has the added advantage that you can simplify the `mount` command. The command in the screenshot below forces `mount` to look for the partition info in `/etc/fstab`.

```
root@linux:~# mount /home/project42
```

33.5. securing mounts

File systems can be secured with several `mount options`. Here are some examples.

33.5.1. ro

The `ro` option will mount a file system as read only, preventing anyone from writing.

```
root@linux ~# mount -t ext2 -o ro /dev/hdb1 /home/project42
root@linux ~# touch /home/project42/testwrite
touch: cannot touch `/home/project42/testwrite': Read-only file system
```

33.5.2. noexec

The `noexec` option will prevent the execution of binaries and scripts on the mounted file system.

```
root@linux ~# mount -t ext2 -o noexec /dev/hdb1 /home/project42
root@linux ~# cp /bin/cat /home/project42
root@linux ~# /home/project42/cat /etc/hosts
-bash: /home/project42/cat: Permission denied
root@linux ~# echo echo hello > /home/project42/helloscript
root@linux ~# chmod +x /home/project42/helloscript
root@linux ~# /home/project42/helloscript
-bash: /home/project42/helloscript: Permission denied
```

33.5.3. nosuid

The `nosuid` option will ignore `setuid` bit set binaries on the mounted file system.

Note that you can still set the `setuid` bit on files.

```
root@linux ~# mount -o nosuid /dev/hdb1 /home/project42
root@linux ~# cp /bin/sleep /home/project42/
root@linux ~# chmod 4555 /home/project42/sleep
root@linux ~# ls -l /home/project42/sleep
-r-sr-xr-x 1 root root 19564 Jun 24 17:57 /home/project42/sleep
```

But users cannot exploit the `setuid` feature.

```
root@linux ~# su - paul
[student@linux ~]$ /home/project42/sleep 500 &
[1] 2876
[student@linux ~]$ ps -f 2876
UID        PID  PPID  C STIME TTY      STAT   TIME CMD
paul      2876  2853  0 17:58 pts/0    S      0:00 /home/project42/sleep 500
[student@linux ~]$
```

33.5.4. noacl

To prevent cluttering permissions with `acl's`, use the `noacl` option.

```
root@linux ~# mount -o noacl /dev/hdb1 /home/project42
```

More `mount options` can be found in the manual page of `mount`.

33.6. mounting remote file systems

33.6.1. smb/cifs

The Samba team (samba.org) has a Unix/Linux service that is compatible with the SMB/CIFS protocol. This protocol is mainly used by networked Microsoft Windows computers.

Connecting to a Samba server (or to a Microsoft computer) is also done with the mount command.

This example shows how to connect to the `10.0.0.42` server, to a share named `data2`.

```
[root@linux ~]# mount -t cifs -o user=paul //10.0.0.42/data2 /home/data2
Password:
[root@linux ~]# mount | grep cifs
//10.0.0.42/data2 on /home/data2 type cifs (rw)
```

The above requires `yum install cifs-client`.

33.6.2. nfs

Unix servers often use `nfs` (aka the network file system) to share directories over the network. Setting up an nfs server is discussed later. Connecting as a client to an nfs server is done with `mount`, and is very similar to connecting to local storage.

This command shows how to connect to the nfs server named `server42`, which is sharing the directory `/srv/data`. The `mount point` at the end of the command (`/home/data`) must already exist.

```
[root@linux ~]# mount -t nfs server42:/srv/data /home/data
[root@linux ~]#
```

If this `server42` has ip-address `10.0.0.42` then you can also write:

```
[root@linux ~]# mount -t nfs 10.0.0.42:/srv/data /home/data
[root@linux ~]# mount | grep data
10.0.0.42:/srv/data on /home/data type nfs (rw,vers=4,addr=10.0.0.42,clienta\
ddr=10.0.0.33)
```

33.6.3. nfs specific mount options

```
bg If mount fails, retry in background.
fg (default)If mount fails, retry in foreground.
soft Stop trying to mount after X attempts.
hard (default)Continue trying to mount.
```

The `soft+bg` options combined guarantee the fastest client boot if there are NFS problems.

```
retrans=X Try X times to connect (over udp).
tcp Force tcp (default and supported)
udp Force udp (unsupported)
```

33.7. practice: mounting file systems

1. Mount the small 200MB partition on /home/project22.

2. Mount the big 400MB primary partition on /mnt, then copy some files to it (everything in /etc). Then umount, and mount the file system as read only on /srv/nfs/salesnumbers. Where are the files you copied ?

3. Verify your work with `fdisk`, `df` and `mount`. Also look in `/etc/mtab` and `/proc/mounts`.

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

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

6. What happens when you mount two file systems on the same mount point ?

7. (optional) Describe the difference between these commands: find, locate, updatedb, makewhatis, whereis, apropos, which and type.

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

33.8. solution: mounting file systems

1. Mount the small 200MB partition on /home/project22.

```
mkdir /home/project22
mount /dev/sdc1 /home/project22
```

2. Mount the big 400MB primary partition on /mnt, then copy some files to it (everything in /etc). Then umount, and mount the file system as read only on /srv/nfs/salesnumbers. Where are the files you copied ?

```
mount /dev/sdb1 /mnt
cp -r /etc /mnt
ls -l /mnt

umount /mnt
ls -l /mnt

mkdir -p /srv/nfs/salesnumbers
mount /dev/sdb1 /srv/nfs/salesnumbers

You see the files in /srv/nfs/salenumbers now...

But physically they are on ext3 on partition /dev/sdb1
```

3. Verify your work with `fdisk`, `df` and `mount`. Also look in `/etc/mtab` and `/proc/mounts`.

```
fdisk -l
df -h
mount

All three the above commands should show your mounted partitions.

grep project22 /etc/mtab
grep project22 /proc/mounts
```

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

```
add the following lines to /etc/fstab

/dev/sdc1 /home/project22 auto defaults 0 0
/dev/sdb1 /srv/nfs/salesnumbers auto defaults 0 0
```

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

```
The files are hidden until umount.
```

6. What happens when you mount two file systems on the same mount point ?

```
Only the last mounted fs is visible.
```

7. (optional) Describe the difference between these commands: find, locate, updatedb, makewhatis, whereis, apropos, which and type.

```
man find
man locate
 ...
```

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

```
# umount /srv/nfs/salesnumbers (optional but recommended)
# fsck /dev/sdb1
```

34. introduction to uuid's

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

A **uuid** or *universally unique identifier* is used to uniquely identify objects. This 128bit standard allows anyone to create a unique *uuid*. That is, the number of *uuid's* that can be generated is so large that the probability of generating a duplicate is extremely low.

This chapter takes a brief look at *uuid*'s.

34.1. lsblk -f

You can quickly locate the `uuid` of file systems with `lsblk -f`. The following example is from a VM running Ubuntu 24.05.

```
student@ubuntu:~$ lsblk -f
NAME    FSTYPE  FSVER  LABEL UUID                                   FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1
├─sda2  ext4    1.0          0e751ccc-2139-4c7a-a90e-e41e9a522aee  1.7G    5% /boot
└─sda3    LVM2_me LVM2 001       d2OsZK-N5Ih-NCA3-TOIv-h9Ul-wXzA-UoQKtz
  └─ubuntu--vg-ubuntu--lv
       ext4    1.0          40b3fedd-d848-4a74-b7ef-f3acac9554ed  25.5G   11% /
```

The same command will also work on recent EL systems, e.g. AlmaLinux 9:

```
[student@el ~]$ lsblk -f
NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1 swap   1          a4814ebe-c0b2-4819-8129-30f32b3e8772            [SWAP]
└─sda2 xfs                303db791-9236-4ac4-a176-e2a033576d89   60.4G     2% /
```

34.2. tune2fs

Use `tune2fs` to find the `uuid` of a file system.

```
student@ubuntu:~$ sudo tune2fs -l /dev/sda2 | grep UUID
Filesystem UUID:          0e751ccc-2139-4c7a-a90e-e41e9a522aee
```

34.3. uuid

There is more information in the manual of `uuid(1)`, a tool that can generate uuid's.

```
[student@el ~]$ sudo dnf install uuid
[student@el ~]$ uuid
c4212384-75ca-11ef-829a-080027c76768
[student@el ~]$ man uuid
```

(On Debian/Ubuntu/Mint, use `sudo apt install uuid`.)

34.4. uuid in /etc/fstab

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

First we use `tune2fs` to find the `uuid`.

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

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

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

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

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

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

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

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

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

34.5. uuid as a boot device

Recent Linux distributions (Debian, Ubuntu, ...) use `grub` with a `uuid` to identify the root file system.

This example shows how a `root=/dev/sda1` is replaced with a `uuid`.

```
title           Ubuntu 9.10, kernel 2.6.31-19-generic
uuid            f001ba5d-9077-422a-9634-8d23d57e782a
kernel          /boot/vmlinuz-2.6.31-19-generic \
root=UUID=f001ba5d-9077-422a-9634-8d23d57e782a ro quiet splash
initrd          /boot/initrd.img-2.6.31-19-generic
```

The screenshot above contains only four lines. The line starting with `root=` is the continuation of the `kernel` line.

RHEL and derived distributions boot from LVM after a default install.

34.6. practice: uuid and filesystems

1. Find the `uuid` of one of your Linux system's partitions with `tune2fs`.
2. Use this `uuid` in `/etc/fstab` and test that it works with a simple `mount`.
3. (optional) Test it also by removing a disk (so the device name is changed). You can edit settings in vmware/Virtualbox to remove a hard disk.
4. Display the `root=` directive in `/boot/grub/menu.lst`.
5. (optional on ubuntu) Replace the `/dev/xxx` in `/boot/grub/menu.lst` with a `uuid` (use an extra stanza for this). Test that it works.

34.7. solution: uuid and filesystems

1. Find the `uuid` of one of your Linux system's partitions with `tune2fs`.

   ```
   student@ubuntu:~$ sudo tune2fs -l /dev/sda2 | grep UUID
   Filesystem UUID:          0e751ccc-2139-4c7a-a90e-e41e9a522aee
   ```

2. Use this `uuid` in `/etc/fstab` and test that it works with a simple `mount`.

   ```
   $ tail -1 /etc/fstab
   UUID=60926898-2c78-49b4-a71d-c1d6310c87cc /home/pro42 ext3 defaults 0 0
   ```

3. (optional) Test it also by removing a disk (so the device name is changed). You can edit settings in vmware/Virtualbox to remove a hard disk.
4. Display the `root=` directive in `/boot/grub/menu.lst`.

   ```
   student@ubuntu:~$ grep root= /boot/grub/menu.lst
   kernel       /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro selinux=1 quiet
   kernel       /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro selinux=1 single
   ```

5. (optional on ubuntu) Replace the `/dev/xxx` in `/boot/grub/menu.lst` with a `uuid` (use an extra stanza for this). Test that it works.

Part XI.

DNS server

35. introduction to DNS

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/, Bert Van Vreckem https://github.com/bertvv/)

DNS is a fundamental part of every large computer network. *DNS* is used by many network services to translate names into network addresses and to locate services on the network (by name).

Whenever you visit a web site, send an e-mail, log on to Active Directory, play Minecraft, chat, or use VoIP, there will be one or (many) more queries to *DNS* services.

For example, when you type https://linux-training.be in a browser address bar, a DNS query is sent to resolve *linux-training.be* to 188.40.26.208. The browser will set up a TCP connection with this IP-address and will use HTTP to obtain the website.

Should *DNS* fail on any level, then the whole network will grind to a halt (unless you hard-coded the network addresses, which is infeasible nowadays).

You will notice that even the largest of organizations benefit greatly from having a *DNS* infrastructure. Thus *DNS* requires all business units to work together.

Even at home, most home modems and routers have builtin *DNS* functionality.

This module will explain what *DNS* actually is and how to interact with a DNS server on a Linux system.

Further reading:

- Wood, Robin. (n.d.) *ZoneTransfer.me*. Retrieved 2014-06-15 from https://digi.ninja/projects/zonetransferme.php
- Nadh, Kalaish. (2022) *DNS Toys*. Retrieved 2022-02-02 from https://www.dns.toys

35.1. about DNS

35.1.1. name to ip address resolution

The *domain name system* or *DNS* is a service on a TCP/IP network that enables clients to translate host names into ip addresses. Actually *DNS* is much more than that, but let's keep it simple for now.

When you use a browser to go to a website, then you type the name of that website in the url bar. But for your computer to actually communicate with the web server hosting said website, your computer needs the ip address of that web server. That is where *DNS* comes in.

In wireshark you can use the *DNS* filter to see this traffic.

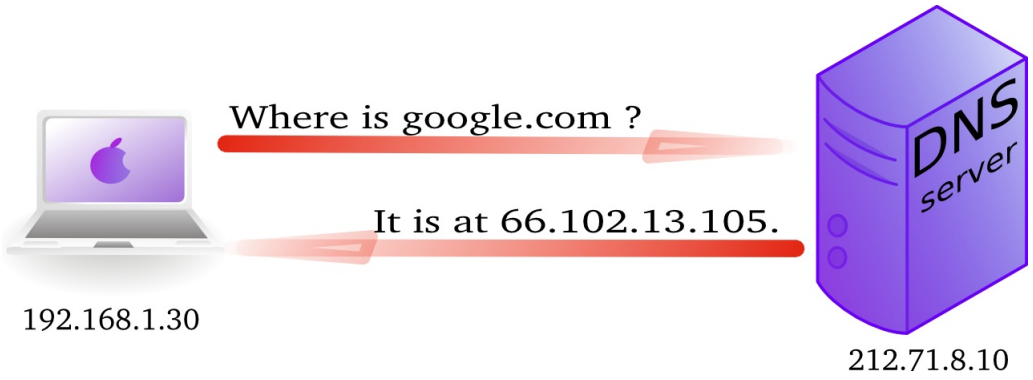


Figure 35.1.: Simplified depiction of a DNS query.

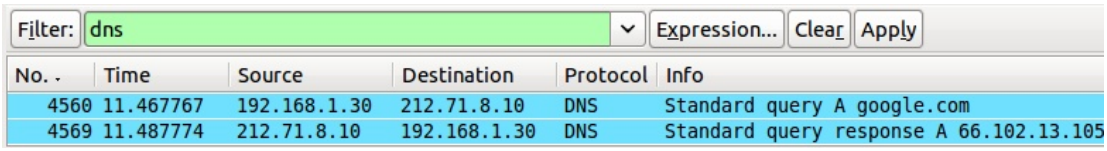


Figure 35.2.: Filtering DNS traffic in Wireshark. Enter `dns` in the Filter text field. The first line shows the request made by the client with IP address 192.168.1.30 and the second the response given by the DNS server at 212.71.8.10.

35.1.2. history

In the Seventies, only a few hundred computers were connected to the internet. To resolve names, computers had a flat file that contained a table to resolve hostnames to ip addresses. This local file was downloaded from `hosts.txt` on an ftp server in Stanford.

In 1984 *Paul Mockapetris* created *DNS*, a distributed treelike hierarchical database that will be explained in detail below.

Today, *DNS* or *Domain Name System* is a worldwide distributed hierarchical database controlled by *ICANN* (the *Internet Corporation for Assigned Names and Numbers*). Its primary function is to resolve names to ip addresses, and to point to internet servers providing `smtp` or `ldap` services.

The old `hosts.txt` file is still active today on most computer systems under the name `/etc/hosts` (or `C:/Windows/System32/Drivers/etc/hosts`). See the `hosts(5)` for details.

A hosts file may look like this:

```
# IP-address    hostname     aliases
127.0.0.1       localhost    localhost.localdomain
::1             localhost6   localhost6.localdomain6

172.22.255.254 router4038   gw gw.netlab.hogent.be
172.22.0.2     server4038   server4038.netlab.hogent.be
172.22.0.3     printer4038 printer4038.netlab.hogent.be
```

Each line contains a “record” with several fields, separated by whitespace. The first field is the IP (IPv4 or IPv6) address, following fields are hostnames and aliases.

You can use the hosts file to define shortcuts or aliases to websites or host names you use often. Contents of the hosts file will override all other means of name resolution.

Just enter the IP address and the chosen hostname in the hosts file and you can use that hostname in your browser, or when interacting with the host on the command line (e.g. `ping gw`, `ssh admin@server4038`, etc.).

A neat trick is to use the hosts file to block network traffic to unwanted hosts, e.g. ad servers or known malware domains. Just point the hostname to the IP address `127.0.0.1` and the website will not be reachable. You can find an example of an elaborate hosts file that blocks hundreds of ad servers at https://someonewhocares.org/hosts/.

35.1.3. DNS namespace

The *dns namespace* is hierarchical tree structure, with the *root servers* (aka dot-servers) at the top. The *root servers* are usually represented by a dot.

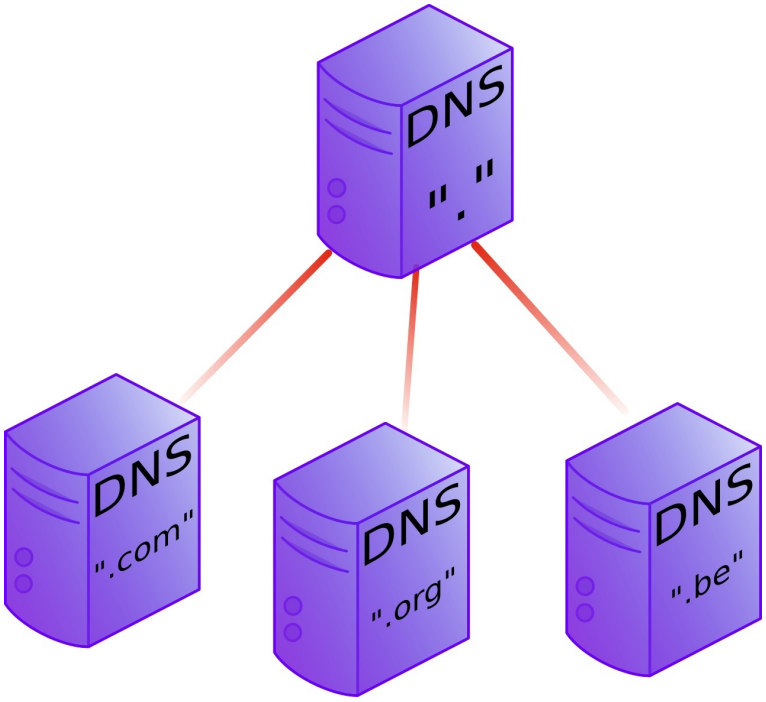


Figure 35.3.: Top-level hierarchy of the DNS namespace.

Below the *root-servers* are the *Top Level Domains* or *TLD*'s. There are more *TLD*'s than shown in the picture. Currently about 200 countries have a *TLD*. (e.g. .be, .nl, .sh, etc.) And there are several general *TLD*'s like .com, .edu, .org, .gov, .net, .mil, .int and more recently also .aero, .info, .museum, ...

35.1.3.1. root servers

There are thirteen *root servers* on the internet, they are named *A* to *M*. Journalists often refer to these servers as *the master servers of the internet*, because if these servers go down, then nobody can (use names to) connect to websites.

The root servers are not thirteen physical machines, there are many more. For example the *F* root server consists of 46 physical machines that all behave as one (using anycast).

For more information, see:

- http://root-servers.org
- http://en.wikipedia.org/wiki/Root_nameserver

35.1.3.2. top level domains

Below the root level are the *top level domains* or *TLD*'s. Originally there were only seven defined:

Year	TLD	Purpose
1985	.arpa	Reverse lookup via in-addr.arpa
1985	.com	Commercial Organizations
1985	.edu	US Educational Institutions
1985	.gov	US Government Institutions
1985	.mil	US Military
1985	.net	Internet Service Providers, Internet Infrastructure
1985	.org	Non profit Organizations
1988	.int	International Organizations like nato.int

Country *TLD*'s were defined for individual countries, like *.uk* in 1985 for the United Kingdom, *.be* for Belgium in 1988 and *.fr* for France in 1986. See RFC 1591 for more info.

In 1998 seven new general purpose *TLD*'s where chosen, they became active in the 21st century.

Year	TLD	Purpose
2002	.aero	Aviation related
2001	.biz	Businesses
2001	.coop	For co-operatives
2001	.info	Informative internet resources
2001	.museum	For museums
2001	.name	For all kinds of names, pseudonyms and labels
2004	.pro	For professionals

Many people were surprised by the choices, claiming not much use for them and wanting a separate *.xxx* domain (introduced in 2011) for adult content, and *.kidz* a save haven for children. In the meantime more *TLD*'s were create like *.travel* (for travel agents), *.tel* (for internet communications) and *.jobs* (for jobs sites).

In 2012 *ICANN* released a list of 2000 new *TLD*'s that would gradually become available. The current list can be found at https://www.iana.org/domains/root/db.

35.1.3.3. domains

One level below the *top level domains* are the *domains*. Domains can have subdomains (also called child domains).

DNS domains are registered at the *TLD* servers, the *TLD* servers are registered at the *dot servers*.

35.1.3.4. fully qualified domain name

The *fully qualified domain name* or *fqdn* is the combination of the *hostname* of a machine appended with its *domain name*.

If for example a system is called *gwen* and it is in the domain *linux-training.be*, then the fqdn of this system is *gwen.linux-training.be*.

On Linux systems you can use the `hostname` and `dnsdomainname` commands to verify this information.

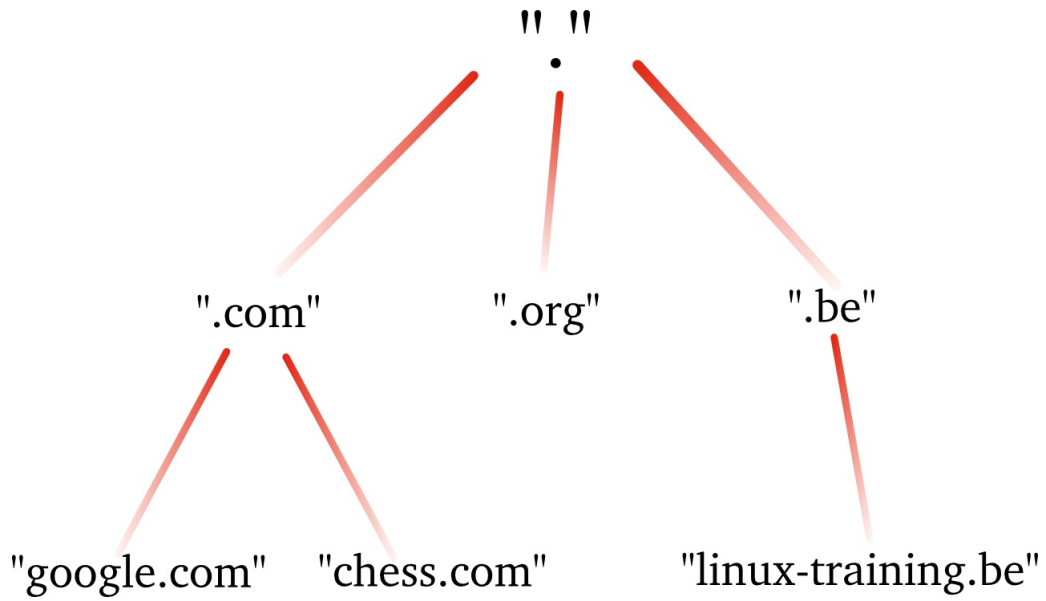


Figure 35.4.: This picture shows *dns domains* like google.com, chess.com, linux-training.be (there are millions more).

```
student@gwen:~$ hostname
gwen
student@gwen:~$ dnsdomainname
linux-training.be
student@gwen:~$ hostname --fqdn
gwen.linux-training.be
```

35.1.3.5. dns zones

A *zone* (aka a *zone of authority*) is a portion of the DNS tree that is covered covers one domain name or child domain name. The picture below represents zones as blue ovals. Some zones will contain delegate authority over a child domain to another zone.

A *dns server* can be *authoritative* over zero, one or more *dns zones*, which means that it is the source of truth for the mapping of names to IP addresses within that/those zone(s). We will see more details later on the relation between a *dns server* and a *dns zone*.

35.1.4. dns records

A *dns zone* is a collection of *records*, also called *resource records* (RRs). This section lists some of those *resource records*.

- The **A record**, which is also called a *host record* contains the ipv4-address of a computer. When a DNS client queries a DNS server for an A record, then the DNS server will resolve the hostname in the query to the specified ip address. An **AAAA record** is similar but contains an ipv6 address instead of ipv4.
- A **PTR record** is the reverse of an A record. It contains the name of a computer and can be used to resolve an ip address to a hostname. This is called a *reverse lookup* or *reverse lookup query*
- A **NS record** or *nameserver record* is a record that points to an authoritative DNS name server (in this zone). You can list all your name servers for your DNS zone in distinct NS records.

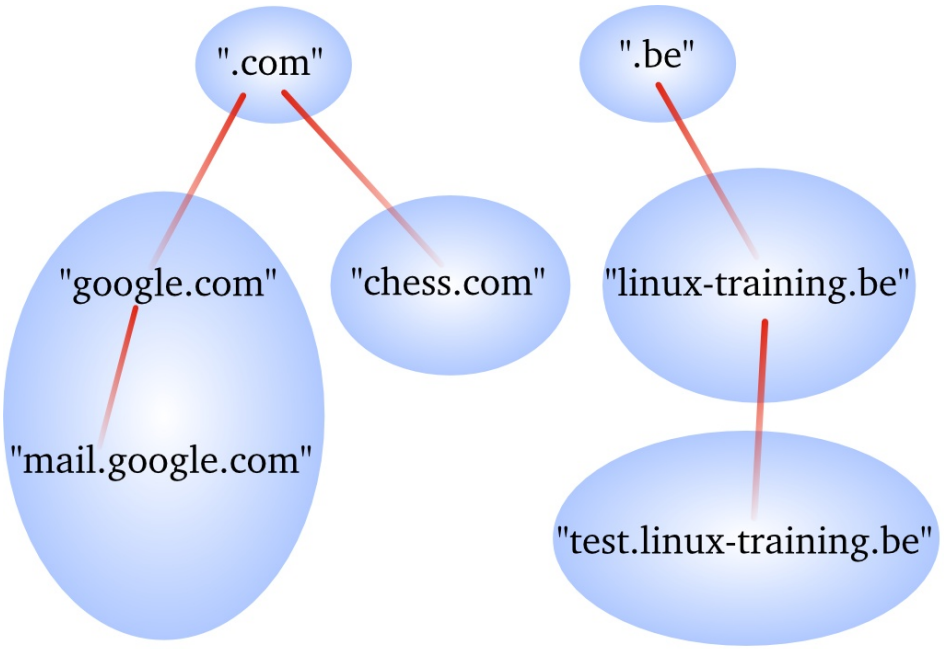


Figure 35.5.: DNS zones of authority.

- An *A record* that maps the name of an *NS record* to an ip address is said to be a **glue record**.
- The **SOA record** (*Start Of Authority*) of a zone contains meta-information about the zone itself. The contents of the SOA record is explained in detail in the section about zone files. There is exactly one SOA record for each zone.
- A **CNAME record** maps a hostname to a hostname, creating effectively an alias for an existing hostname. The name of the mail server is often aliased to *mail* or *smtp*, and the name of a web server to *www*.
- The **MX record** points to an *smtp server*. When you send an email to another domain, then your mail server will need the MX record of the target domain's mail server in order to deliver email to the recepient's mailbox.

35.2. DNS queries

The question a client asks a dns server is called a *query*. When a client queries for an ip address, this is called a *forward lookup query* (as seen in the previous drawing). The reverse, a query for the name of a host, is called a *reverse lookup query*.

We'll show in the following sections how to perform DNS queries with several command line tools.

This is what a reverse lookup looks like when sniffing with `tcpdump` (note the occurrecnes of PTR in the output):

```
student@linux:~$ sudo tcpdump udp port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:01:29.357685 IP 192.168.1.103.42041 > 192.168.1.42.domain: 14763+ PT\
R? 87.155.93.188.in-addr.arpa. (44)
11:01:29.640093 IP 192.168.1.42.domain > 192.168.1.103.42041: 14763 1/0\
/0 PTR antares.ginsys.net. (76)
```

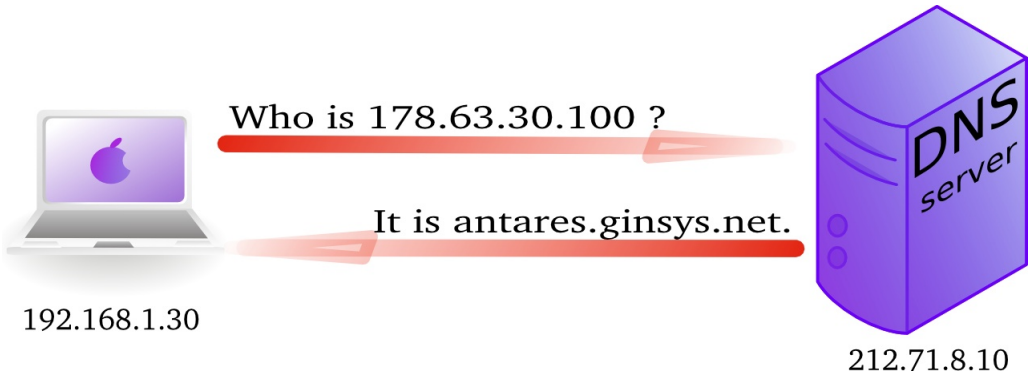


Figure 35.6.: Simplified depiction of a reverse DNS lookup query.

And here is what it looks like in `wireshark` (note this is an older screenshot).

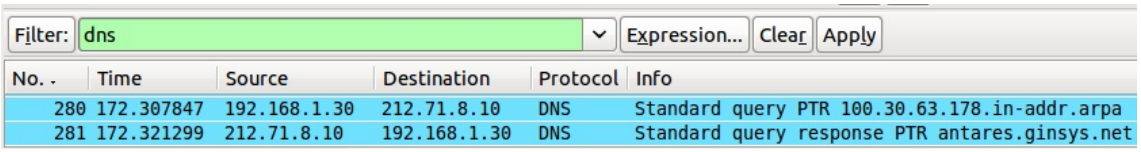


Figure 35.7.: Reverse lookup in Wireshark.

35.2.1. iterative or recursive query

A **recursive query** is a DNS query where the client that is submitting the query expects a complete answer (Like the fat red arrow above going from the Macbook to the DNS server).

An **iterative query** is a DNS query where the client does not expect a complete answer. Iterative queries usually take place between name servers, e.g. asking a root server for the authoritative name server of a top level domain. The root name servers do not respond to recursive queries.

35.3. interacting with DNS

There are several tools to interact with *DNS*. In this section, we'll discuss `host`, `nslookup`, `dig`, and `getent`.

The `ping` command could also be used to test whether DNS resolution is working, but it does additional things like sending ICMP echo requests. Therefore, we will avoid using it for the purpose of testing DNS.

On Enterprise Linux and derivatives, ensure the `bind-utils` package is installed in order to use `dig`, `host`, or `nslookup`. On Debian and derivatives, you'll need the `bind9-host` (for `host`) and `bind9-dnsutils` packages (for `dig` and `nslookup`).

When you're on a system that doesn't have these tools installed, and installing additional packages is not possible, `getent` can be used as a fallback, albeit with limited functionality. Getent is included in the `glibc` library, which is installed on all Linux systems (`glibc-common` on EL and `libc-bin` on Debian).

35.3.1. which DNS server is used?

A client computer needs to know the ip address of the *dns server* to be able to send queries to it. This is either provided by a *dhcp server* or manually entered. See also the chapter on network configuration for more details.

The traditional location to keep this information on Linux systems is in the `/etc/resolv.conf` file.

```
student@linux:~$ cat /etc/resolv.conf
domain linux-training.be
search linux-training.be
nameserver 192.168.1.42
```

If the nameserver address points to 127.0.0.53, then the system is using *systemd-resolved* to resolve names. In this case, you can use the `resolvectl dns` command to see the current configuration, e.g.:

```
student@linux:~$ resolvectl dns
Global:
Link 2 (eth0): 10.0.2.3
Link 3 (eth1):
```

You can manually change the ip address in this file to use another *DNS* server. For example Google provides a public name server at 8.8.8.8 and 8.8.4.4.

```
student@linux:~$ sudo nano /etc/resolv.conf
[ ... edit the file ... ]
student@linux:~$ cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

If your system uses *systemd-resolved*, you should use the `resolvectl dns LINK SERVER` command to change the configuration.

```
student@linux:~$ resolvectl dns
Global:
Link 2 (eth0): 10.0.2.3
Link 3 (eth1):
vagrant@linux:~$ sudo resolvectl dns eth0 8.8.8.8
vagrant@linux:~$ resolvectl dns
Global:
Link 2 (eth0): 8.8.8.8
Link 3 (eth1):
```

Please note that on *dhcp clients* this configuration value can be overwritten when the *dhcp lease* is renewed. Permanent changes to the configuration are discussed in the chapter on network configuration.

35.3.2. getent ahosts

In case when you don't have `dig`, `host`, or `nslookup` available, and installing packages is not feasible, you can use `getent ahosts` to perform a DNS lookup. The `getent` command is used to get entries from the system databases (see the `getent(1)` man page), and `ahosts` is one of the databases it can query.

```
student@linux:~$ getent ahosts linux-training.be
188.40.26.208   STREAM linux-training.be
188.40.26.208   DGRAM
188.40.26.208   RAW
2a01:4f8:d0a:1044::2 STREAM
2a01:4f8:d0a:1044::2 DGRAM
2a01:4f8:d0a:1044::2 RAW
```

The command has no fancy options, but it can be used to perform a forward lookup to the default name server.

35.3.3. host

`host` is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa.

```
student@linux:~$ host linux-training.be
linux-training.be has address 188.40.26.208
linux-training.be has IPv6 address 2a01:4f8:d0a:1044::2
linux-training.be mail is handled by 10 www115.your-server.de.
student@linux:~$ host linux-training.be 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:

linux-training.be has address 188.40.26.208
linux-training.be has IPv6 address 2a01:4f8:d0a:1044::2
linux-training.be mail is handled by 10 www115.your-server.de.
student@linux:~$ host 188.40.26.208
208.26.40.188.in-addr.arpa domain name pointer www115.your-server.de.
student@linux:~$ host 2a01:4f8:d0a:1044::2
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.4.0.1.a.0.d.0.8.f.4.0.1.0.a.2.ip6.arpa domain name poin
server.de
```

The host command is quite limited in its functionality, `nslookup` and especially `dig` are much more useful to interact with DNS.

35.3.4. nslookup

Windows users are probably familiar with the `nslookup` command and will be happy to know that it is also available on Linux systems.

In the first example below, the default DNS server (mentioned in `/etc/resolv.conf` or by `resolvectl dns`) is queried.

```
student@linux:~$ nslookup linux-training.be
Server:         10.0.2.3
Address:        10.0.2.3#53

Non-authoritative answer:
Name:   linux-training.be
Address: 188.40.26.208
Name:   linux-training.be
Address: 2a01:4f8:d0a:1044::2
```

In the following example, we query a specific DNS server, in this case Google's public DNS server at 8.8.8.8. You will need this version of the command when you are troubleshooting issues with a DNS server you're configuring yourself (of course using its IP address instead of 8.8.8.8).

```
student@linux:~$ nslookup linux-training.be 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   linux-training.be
Address: 188.40.26.208
Name:   linux-training.be
Address: 2a01:4f8:d0a:1044::2
```

Reverse lookups are also possible with `nslookup`.

```
student@linux:~$ nslookup 188.40.26.208
208.26.40.188.in-addr.arpa      name = www115.your-server.de.

Authoritative answers can be found from:

student@linux:~$ nslookup 2a01:4f8:d0a:1044::2
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.4.0.1.a.0.d.0.8.f.4.0.1.0.a.2.ip6.arpa      name = www11
server.de.

Authoritative answers can be found from:
```

`nslookup` can also be used interactively:

```
nslookup
> server 8.8.8.8
Default server: 8.8.8.8
Address: 8.8.8.8#53
> linux-training.be
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   linux-training.be
Address: 188.40.26.208
Name:   linux-training.be
Address: 2a01:4f8:d0a:1044::2
> set type=PTR
> 188.40.26.208
208.26.40.188.in-addr.arpa      name = www115.your-server.de.
```

```
Authoritative answers can be found from:
> 2a01:4f8:d0a:1044::2
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.4.0.1.a.0.d.0.8.f.4.0.1.0.a.2.ip6.arpa      name = www11
server.de.

Authoritative answers can be found from:
> set type=NS
> linux-training.be
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
linux-training.be       nameserver = ns.second-ns.com.
linux-training.be       nameserver = ns1.your-server.de.
linux-training.be       nameserver = ns3.second-ns.de.

Authoritative answers can be found from:
> set type=MX
> linux-training.be
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
linux-training.be       mail exchanger = 10 www115.your-server.de.

Authoritative answers can be found from:
> exit
```

35.3.5. dig

The `dig` command is a powerful tool for querying DNS servers. If you manage a DNS server, or need to troubleshoot DNS issues, it is a good idea to get familiar with `dig`, as it is much more versatile than `nslookup`.

```
student@linux:~$ dig linux-training.be
; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> linux-training.be
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8729
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;linux-training.be.             IN      A

;; ANSWER SECTION:
linux-training.be.      7200    IN      A       188.40.26.208

;; Query time: 44 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Mar 27 17:02:44 UTC 2024
;; MSG SIZE  rcvd: 62
```

The output of `dig` is quite verbose, but very informative, and follows the syntax of a BIND zone file (see below). The `;` character is used to indicate comments. In this example, it is indicated that there was one `QUERY` and one `ANSWER`. The `QUESTION SECTION` shows the query that was made, and the `ANSWER SECTION` shows the response. Sometimes, the `ADDITIONAL SECTION` contains glue records for the name servers that are authoritative for the queried domain. Finally, you also get information about the query time, the server that was queried, and the time the query was made.

The `+short` option can be used to get a more concise output.

```
student@linux:~$ dig +short linux-training.be
188.40.26.208
```

If you want to query a specific DNS server, you can specify it with the `@` symbol:

```
student@linux:~$ dig +short @1.1.1.1 linux-training.be
188.40.26.208
```

Another type of *Resource Record* can be specified by just mentioning the type in the query:

```
student@linux:~$ dig +short @1.1.1.1 AAAA linux-training.be
2a01:4f8:d0a:1044::2
student@linux:~$ dig +short NS hogent.be
ens1.hogent.be.
ns3.belnet.be.
ens2.hogent.be.
ns2.belnet.be.
ns1.belnet.be.
student@linux:~$ dig +short MX hogent.be
0 hogent-be.mail.protection.outlook.com.
```

A reverse lookup can be done with the -x option:

```
student@linux:~$ dig -x 188.40.26.208 +short
www115.your-server.de.
```

A useful query is the `ANY` query, which will return all records for a domain:

```
student@linux:~$ dig @ens1.hogent.be ANY hogent.be

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> @ens1.hogent.be ANY hogent.be
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25886
;; flags: qr aa rd; QUERY: 1, ANSWER: 12, AUTHORITY: 0, ADDITIONAL: 5
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 0d2f566a6341135d5ebd6167660452e7ffc6c1184db1c14a (good)
;; QUESTION SECTION:
;hogent.be.                     IN      ANY

;; ANSWER SECTION:
hogent.be.          3600   IN    SOA    ens1.hogent.be. hostmaster.hogent.be. 2024032506 43
```

```
hogent.be.              3600    IN      A       193.190.173.132
hogent.be.            3600    IN     TXT     "adobe-idp-site-verification=...."
hogent.be.              3600    IN      TXT     "docusign=...."
hogent.be.                  3600     IN        TXT      "v=spf1 a:spf-mail-
out.hogent.be include:spf.protection.outlook.com -all"
hogent.be.              3600    IN      TXT     "docusign=...."
hogent.be.              3600    IN      NS      ens2.hogent.be.
hogent.be.              3600    IN      NS      ns1.belnet.be.
hogent.be.              3600    IN      NS      ns3.belnet.be.
hogent.be.              3600    IN      NS      ens1.hogent.be.
hogent.be.              3600    IN      NS      ns2.belnet.be.
hogent.be.                       3600       IN          MX          0  hogent-
be.mail.protection.outlook.com.

;; ADDITIONAL SECTION:
ens1.hogent.be.         3600    IN      A       193.190.172.1
ens2.hogent.be.         3600    IN      A       193.190.172.4
ens1.hogent.be.         3600    IN      AAAA    2001:6a8:1c60:d000::100
ens2.hogent.be.         3600    IN      AAAA    2001:6a8:1c60:d000::4

;; Query time: 24 msec
;; SERVER: 193.190.172.1#53(ens1.hogent.be) (TCP)
;; WHEN: Wed Mar 27 17:09:48 UTC 2024
;; MSG SIZE  rcvd: 674
```

You can request a zone transfer with the AXFR query type. Remark that this will usually not work for public DNS servers, as they are configured to deny zone transfers to the public. When you set up a primary and secondary nameserver yourself, the secondary nameserver will request a zone transfer from the primary nameserver.

The example below shows a zone transfer from `zonetransfer.me`, set up specifically for demonstration purposes. It contains examples for many different types of resource records. See https://zonetransfer.me/ for more information.

```
student@linux:~$ dig AXFR zonetransfer.me @nsztm1.digi.ninja

; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> AXFR zonetransfer.me @nsztm1.digi.ninja
;; global options: +cmd
zonetransfer.me.     7200   IN    SOA    nsztm1.digi.ninja. robin.digi.ninja. 2019100801 1
zonetransfer.me.        300     IN      HINFO   "Casio fx-700G" "Windows XP"

... more lines of output ...

zonetransfer.me.     7200   IN    SOA    nsztm1.digi.ninja. robin.digi.ninja. 2019100801 1
;; Query time: 24 msec
;; SERVER: 81.4.108.41#53(nsztm1.digi.ninja) (TCP)
;; WHEN: Wed Mar 27 17:16:22 UTC 2024
;; XFR size: 50 records (messages 1, bytes 2085)
```

If you don't specify an argument, `dig` will query the *dot domain* `.` and return a list with the names of the root DNS servers:

```
student@linux:~$ dig
; <<>> DiG 9.16.23-RH <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25274
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;.                  IN   NS

;; ANSWER SECTION:
.            20603   IN   NS   h.root-servers.net.
.            20603   IN   NS   b.root-servers.net.
.            20603   IN   NS   g.root-servers.net.
.            20603   IN   NS   l.root-servers.net.
.            20603   IN   NS   a.root-servers.net.
.            20603   IN   NS   k.root-servers.net.
.            20603   IN   NS   j.root-servers.net.
.            20603   IN   NS   e.root-servers.net.
.            20603   IN   NS   d.root-servers.net.
.            20603   IN   NS   i.root-servers.net.
.            20603   IN   NS   m.root-servers.net.
.            20603   IN   NS   c.root-servers.net.
.            20603   IN   NS   f.root-servers.net.

;; Query time: 19 msec
;; SERVER: 10.0.2.3#53(10.0.2.3)
;; WHEN: Wed Mar 27 21:23:11 UTC 2024
;; MSG SIZE  rcvd: 239
```

As mentioned before, each of the thirteen root servers has multiple instances located all over the world. You can ask a root name server to reveal its actual name that should contain an IATA three letter airport code.

```
student@linux:~$ dig +short +norec @f.root-servers.net hostname.bind chaos txt
"BRU.cf.f.root-servers.org"
```

35.4. practice: dns

Log in to a Linux system and try the following:

1. What's the IP address for the DNS server for this system? How did you find it? This may differ depending on your Linux distribution.
2. Try some DNS queries using `getent`, `host`, `nslookup` and `dig`. See how the output differs.
   - Try to resolve the IP address for `www.linux-training.be` or some well known websites.
   - Try reverse DNS lookups (PTR) and other types of queries (AAAA, MX, SOA, etc.).
3. Try to determine the authoritative DNS server for your Internet Service Provider, or for the DNS server of the organization you are affiliated with.
   - Try some forward (IPv4 and IPv6) and reverse DNS queries.
   - Try to find the mail server for this domain
   - The following exercise may yield different results depending whether you're inside the organization's network or not. Can you request a zone transfer? What happens? If you do get a zone transfer, what information do you get?
4. Go to https://www.dns.toys and try some of the tools there.
5. Go to https://digi.ninja/projects/zonetransferme.php and try the suggested exercises there.

35.5. solution: dns

TODO

36. the BIND DNS server

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/, Bert Van Vreckem https://github.com/bertvv/)

In this module, we'll discuss how to set up ISC BIND, the most widely used implementation of DNS on a Linux system.

Learning goals:

- Install BIND
- Configure BIND as a caching or forwarding name server
- Configure BIND as an authoritative name server
- Configure BIND as a secondary name server

Further reading:

- Aitchison, Ron. *DNS for Rocket Scientists*. http://www.zytrax.com/books/dns/
- Mens, Jan-Piet. *Alternative DNS Servers*. UIT Cambridge, 2008.

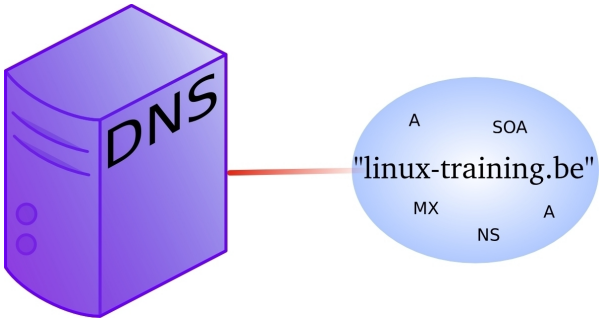
36.1. DNS server types

There are several types of DNS servers, each with its own purpose. The most common types are:

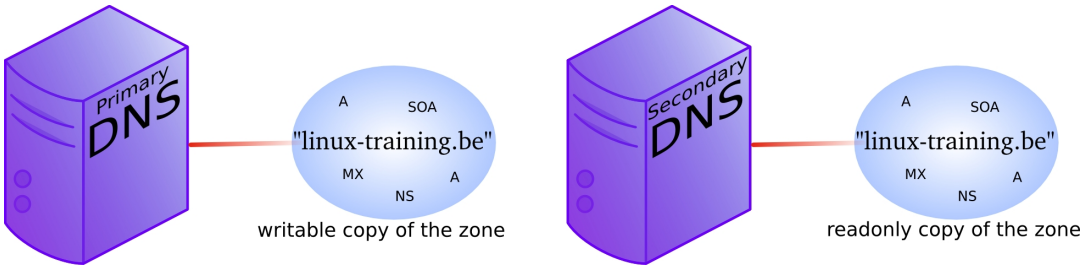
36.1.1. Authoritative name server

This type of server is the "source of truth" for a specific *DNS zone*. When a query is made for a record in the zone, the authoritative server returns the answer. **Authoritative** servers can be **primary** or **secondary**[1] **servers**. A *primary server* is the first authoritative DNS server for a domain, and it has a (human readable) *zone file* that contains all the resource records for the zone.

For reasons of fault tolerance, performance or load balancing you may decide to set up another *DNS server* with authority over that zone. This is called a *secondary* dns server. A *secondary server* replicates the zone data from the primary server through a *zone transfer* (and stores it in a binary format).



[1] The old nomenclature of *master* and *slave* servers is being phased out due to its negative connotations. The terms *primary* and *secondary* are now preferred.



36.1.2. Caching name server

This type of server is not authoritative, but passes on the query to another server and then caches responses to queries it receives.

When a query is made, the caching server first checks its cache to see if it has the answer. If it does, it returns the cached response. If not, it starts an **iterative query.** In an iterative query, the resolver will query a root server for the DNS server responsible for the top-level domain of the query. It will then query that server for the authoritative DNS server for the second-level domain, and so on, until it gets the answer to the original query. The caching server will cache the response to the original query for a certain amount of time (the *TTL* or *Time To Live* value of the record). Subsequent queries for the same record will be answered from the cache.

For example, a client queries for the A record on *www.linux-training.be* to its local server stored in `/etc/resolv.conf`. This is the first query ever received by this local server. The local server checks that it is not authoritative for the linux-training.be domain, nor for the *.be TLD*, and it is also not a root server. So the local server will use the root hints to send an *iterative* query to a root server.

The root server will reply with a reference to the server that is authoritative for the .be domain (root DNS servers do not resolve fqdn's, and root servers do not respond to recursive queries).

The local server will then send an iterative query to the authoritative server for the *.be tld*. This server will respond with a reference to the name server that is authoritative for the *linux-training.be* domain.

The local server will then sent the query for *www.linux-training.be* to the authoritative server (or one of its slave servers) for the *linux-training.be* domain. When the local server receives the ip address for *www.linux-training.be*, then it will provide this information to the client that submitted this query.

Besides caching the A record for *www.linux-training.be*, the local server will also cache the NS and A record for the *linux-training.be* name server and the .be name server.

36.1.3. Forwarding name server

A **Forwarding name server** is not authoritative, but it forwards queries to other servers. The DNS server of a VirtualBox NAT interface is an example of a forwarding server that passes on queries of the VM to the DNS server of the physical machine.

36.1.4. Stealth name server

This type of server is hidden from the public and is used for internal purposes. It is not listed in the publicly visible NS records for a domain. **Stealth servers** are used for DNS resolution of network services for internal use and are not accessible from the internet.

For example, a company might have an intranet webserver with hostname `intranet.company.com` that is only accessible from within the company network. The RRs for this host would only be available from the internal stealth DNS server, but not from the public authoritative DNS server for `company.com`.

36.1.5. Split horizon server

This type of server provides different responses to queries based on the source of the query. For example, a **split horizon server** might return different IP addresses for a domain based on whether the query is coming from inside or outside the local network.

36.1.6. Best practices

A DNS server, depending on how it is configured, can have one or more properties from the above list. For example, a server can be both authoritative and caching, it can be caching and forwarding, or both primary and secondary (for different domains).

Some best practices for maintaining a DNS server in a production environment:

- *Don't configure your DNS server to support recursive queries for all clients*. This could lead to DDoS attacks. You can always limit the source IP addresses to trusted clients that are allowed to make recursive queries.
- *A secure DNS server should only perform a single function*. For example, a DNS server that is authoritative for a zone should not also be a caching server (this is called an **authoritative only server**). Unfortunately, this is not always feasible for smaller organizations and in practice, many DNS servers perform multiple functions.
- *A DNS server should be installed on a dedicated machine* because it is essential for the operation of a network. Combining a DNS server with other services on the same machine can lead to lower availability (if the services are under high load) or security issues (since it increases the attack surface).

36.1.7. caching only servers

A *dns server* that is set up without *authority* over a *zone*, but that is connected to other name servers and caches the queries is called a *caching only name server*. Caching only name servers do not have a *zone database* with resource records. Instead they connect to other name servers and cache that information.

There are two kinds of caching only name servers. Those with a *forwarder*, and those that use the *root servers*.

The default installation of BIND is a caching ony name server without a forwarder.

36.1.7.1. caching only server without forwarder

A caching only server without forwarder will have to get information elsewhere. When it receives a query from a client, then it will determine the response through a series of *iterative queries* (as described earlier). In the end, our hard working *DNS* server will find an answer and report this back to the client.

In the picture below, the clients asks for the ip address of *linux-training.be*. Our caching only server will contact the root server, and be refered to the *.be* server. It will then contact the *.be* server and be refered to one of the name servers of Openminds. One of these name servers (in this cas *ns1.openminds.be*) will answer the query with the ip address of *linux-training.be*.

When our caching only server reports this to the client, then the client can connect to this website.

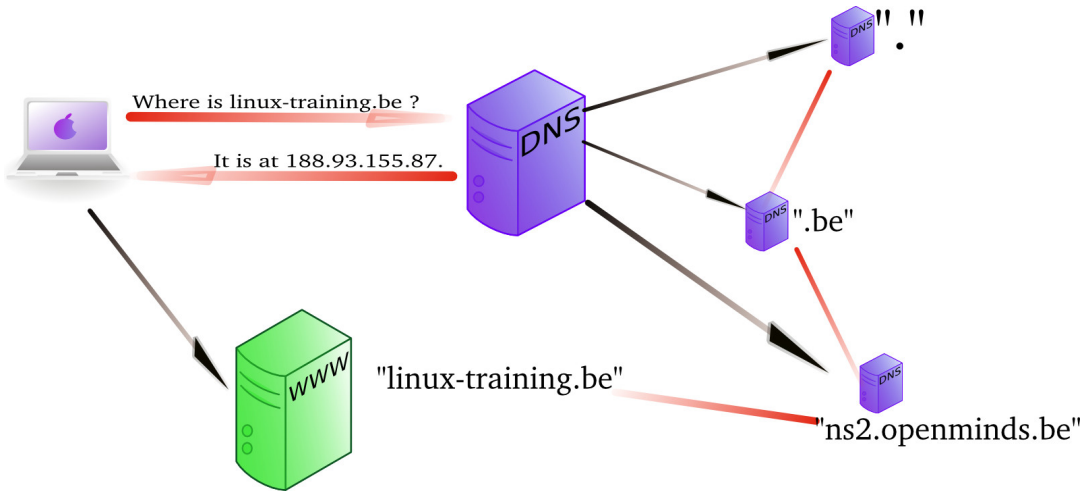


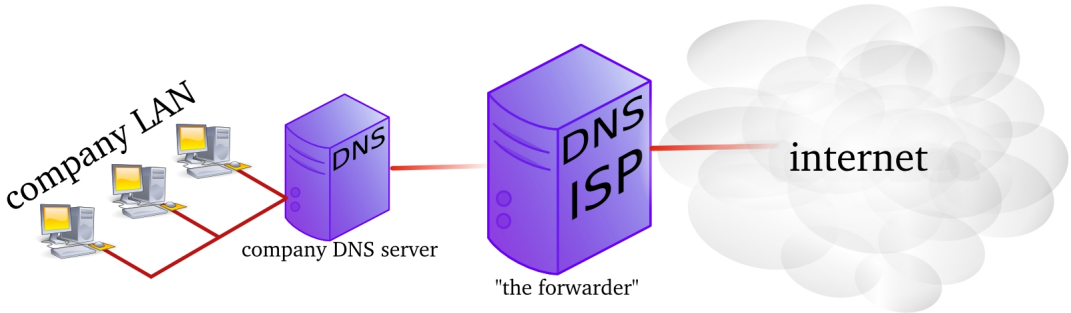
Figure 36.1.: Caching name server resolving *linux-training.be*.

Sniffing with `tcpdump` will give you this (the first 20 characters of each line are cut).

```
192.168.1.103.41251 > M.ROOT-SERVERS.NET.domain: 37279% [1au] A? linux-tr\
aining.be. (46)
M.ROOT-SERVERS.NET.domain > 192.168.1.103.41251: 37279- 0/11/13 (740)
192.168.1.103.65268 > d.ns.dns.be.domain: 38555% [1au] A? linux-training.\
be. (46)
d.ns.dns.be.domain > 192.168.1.103.65268: 38555- 0/7/5 (737)
192.168.1.103.7514 > ns2.openminds.be.domain: 60888% [1au] A? linux-train\
ing.be. (46)
ns2.openminds.be.domain > 192.168.1.103.7514: 60888*- 1/0/1 A 188.93.155.\
87 (62)
```

36.1.7.2. caching only server with forwarder

A *caching only server* with a *forwarder* is a DNS server that will get all its information from the *forwarder*. The *forwarder* must be a *dns server* for example the *dns server* of an *internet service provider*.



This picture shows a *dns server* on the company LAN that has set the *dns server* from their *isp* as a *forwarder*. If the ip address of the *isp dns server* is 212.71.8.10, then the following lines would occur in the `named.conf` file of the company *dns server*:

```
forwarders {
    212.71.8.10;
};
```

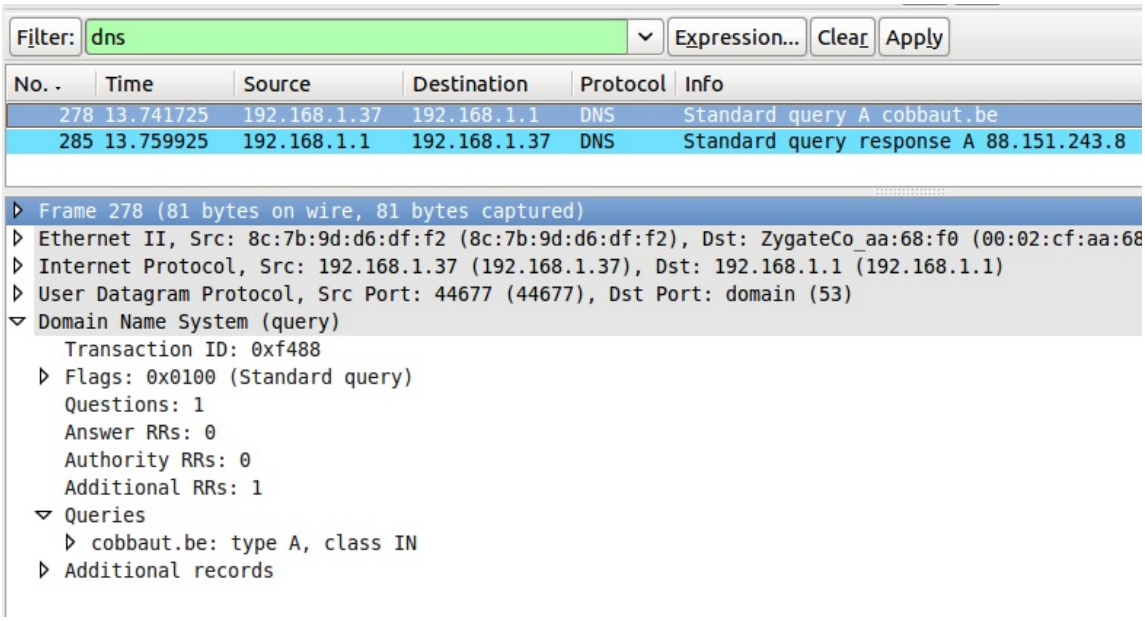


Figure 36.2.: Example of a forwarded DNS query, captured by Wireshark.

You can also configure your DNS server to work with *conditional forwarder(s)*. The definition of a conditional forwarder looks like this.

```
zone "someotherdomain.local" {
        type forward;
        forward only;
        forwarders { 10.104.42.1; };
};
```

36.2. BIND installation

If you need to set up a DNS server, there are several open source solutions available. PowerDNS, MyDNS, MaraDNS, dnsmasq, Name Server Daemon (NSD), Unbound, etc. However, the most widely used DNS server on the internet is undoubtedly BIND (Berkeley Internet Name Domain). BIND is open source and maintained by the *Internet Systems Consortium* (ISC).

In this section, we'll discuss how to install and configure BIND on a Linux system.

36.2.1. installation on Debian

On **Debian-based systems**, install the `bind9` package:

```
student@debian:~$ sudo apt install bind9
```

The service will be started automatically after installation. You can check the status of the service with `systemctl status named`.

The main configuration files for BIND can be found in the `/etc/bind` directory and are named `named.conf*`. The `named.conf` file is the main configuration file for BIND, and it includes other configuration files. The `named.conf.options` file contains options that apply to the entire server, such as the listening interfaces and the forwarders. The `named.conf.local`

file contains the zone definitions for the server. The `named.conf.default-zones` file contains the *default zones* (e.g. the root, broadcast and localhost zones) that are included in the configuration.

Zone files are kept in the same directory and are named after the zone they represent. By default, you will find several files starting with `db.*` and `zones.rfc1918`.

36.2.2. installation on Enterprise Linux

On **Enterprise Linux**, install the `bind` package:

```
student@el:~$ sudo dnf install bind
```

The service will *not* be started automatically, so you will need to start it manually and enable it to start at boot:

```
student@el:~$ sudo systemctl enable --now named
```

For security, BIND configuration files and directories are only readable for the root user, so you will need to use `sudo` to view or edit them (or, optionally, become `root`). The main configuration file is `/etc/named.conf`, and the zone files are kept in the `/var/named` directory.

36.2.3. comparison between Debian and Enterprise Linux installation

Although BIND on EL and Debian are the same software, there are considerable differences in the way they are installed and configured. The config files are stored in a different location and are structured differently.

The main differences are:

	Debian	EL
Package name	`bind9`	`bind`
Service name	`named`	`named`
Configuration directory	`/etc/bind`	`/etc`, `/etc/named`
Main config file	`/etc/bind/named.conf`	`/etc/named.conf`
Server options	`/etc/bind/named.conf.options`	`/etc/named.conf`
Default zone definitions	`/etc/bind/named.conf.default-zones`	`/etc/named.conf`
Zone files directory	`/etc/bind`	`/var/named`
Root hints file	`/usr/share/dns/root.hints`	`/var/named/named.ca`
Default zone files	`/etc/bind/db.*`, `/etc/bind/zones.rfc1918`	`/var/named/named.*`

36.2.4. troubleshooting commands

Before we start configuring BIND, we'll first introduce some useful commands to observe how BIND works, that can be used for troubleshooting configuration issues. We'll show the commands on an EL system, but they work on Debian as well.

Checking the status of the BIND service:

```
[student@el ~]$ systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
    Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: disabled)
     Active: active (running) since Sat 2024-06-15 19:44:21 UTC; 3min 32s ago
   Process: 4812 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING" == "yes" ]; then /
checkconf -z "$NAMEDCONF"; else echo "Checking of zone files is disabled"; fi (c>
   Process: 4815 ExecStart=/usr/sbin/named -u named -c ${NAMEDCONF} $OPTIONS (code=exited, s
    Main PID: 4816 (named)
       Tasks: 10 (limit: 11128)
     Memory: 25.2M
        CPU: 44ms
     CGroup: /system.slice/named.service
             └─4816 /usr/sbin/named -u named -c /etc/named.conf
```

Checking interfaces and network ports that BIND is listening on:

```
[student@el ~]$ sudo ss -tlnp | grep named
State   Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
LISTEN  0     10       127.0.0.1:53      0.0.0.0:*    users:(("named",pid=4816,fd=34))
LISTEN  0     10       127.0.0.1:53      0.0.0.0:*    users:(("named",pid=4816,fd=35))
LISTEN  0     4096      127.0.0.1:953     0.0.0.0:*    users:(("named",pid=4816,fd=31))
LISTEN  0     4096        [::1]:953       [::]:*    users:(("named",pid=4816,fd=42))
LISTEN  0     10          [::1]:53        [::]:*    users:(("named",pid=4816,fd=41))
LISTEN  0     10          [::1]:53        [::]:*    users:(("named",pid=4816,fd=40))
```

So immediately after installation, BIND is only listening on the loopback interface, on port 53 for DNS queries and on port 953 for `rndc` commands. Both IPv4 and IPv6, TCP and UDP are supported. UDP is typically used for simple DNS queries, while TCP is used for zone transfers and large responses.

Checking whether the configuration file is correct:

```
[student@el ~]$ sudo named-checkconf
```

If the syntax of the main configuration file is correct, the command will return without any output and exit status 0. If there are errors, they will be displayed on the screen and the process will exit with a non-zero exit status.

Checking the syntax of a zone file is done with `named-checkzone <zone> <zonefile>`. Later, we'll create our own zone files, but the following command can be run on a basic installation, and checks the root hints file. On Debian, you'll need to change the path to the root hints file.

```
[student@el ~]$ sudo named-checkzone . /var/named/named.ca
zone ./IN: has 0 SOA records
zone ./IN: not loaded due to errors.
```

You should check the syntax of the configuration and zone files after each change to ensure that the server will (re)start correctly.

Checking the logs of the BIND server with `journalctl`:

```
[student@el ~]$ sudo journalctl -u named.service
... output omitted ...
```

When troubleshooting a BIND server, it's useful to keep the logs open in a separate terminal. Add the `-f` option to the `journalctl` command to follow the logs in real-time and `-l` to wrap long lines:

```
[student@el ~]$ sudo journalctl -flu named.service
... output omitted ...
```

Finally, you can enable and disable logging of queries with `rndc`:

```
[student@el ~]$ sudo rndc querylog on
[student@el ~]$ sudo rndc querylog off
```

The command doesn't return any output, but you'll find the result in the logs. If you don't add the option `on` or `off`, the current status will be toggled.

36.3. main BIND configuration file

The main BIND configuration file `named.conf` usually consists of several sections. The most important sections are:

- `options`: configure interfaces, recursion, DNSSEC, etc.
- `logging`: configure logging
- `zone`: define zones
- include other configuration files

In the sections below, we show how to change basic configuration options for a BIND server. Remark that this is not a comprehensive guide to BIND configuration. BIND is a very powerful and flexible DNS server, and there are many more options available. For a more elaborate explanation, check e.g. DNS for Rocket Scientists or the BIND documentation.

36.3.1. control who can query the server

In a default installation, BIND can not be queried by other hosts, even though it is running. In order to make the service available over the network, you need to configure the *network interfaces* that BIND listens on, and determine which hosts are allowed to query the server.

To configure the interfaces that BIND listens on, you can use the `listen-on` directive. By default, BIND listens only on the loopback interface. Change this to the IP address(es) of the interface(s) you want BIND to listen on, or `any` to listen on all interfaces. Each IP address should be terminated with a semicolon.

The default configuration in the `options` section may look like:

```
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
```

Change this to e.g.:

```
    listen-on port 53 { any; };
    listen-on-v6 port 53 { any; };
```

The hosts that can send queries to the server are configured with the `allow-query` directive. By default, BIND only allows queries from localhost. To allow queries from any host, change the directive to:

```
    allow-query { any; };
```

You can also specify a range of IP addresses (e.g. 192.168.56/24), a single IP address, special keywords like `localhost`, `localnets`, `any`, or a semicolon-separated list of these.

After making the change, check the syntax of the configuration file and restart the service:

```
[student@el ~]$ sudo named-checkconf
[student@el ~]$ sudo systemctl restart named
```

36.3.2. recursion

Whether your name server is allowed to perform recursive queries depends on the `recursion` directive in the `options` section.

```
    recursion yes;
```

If you set up an authoritative server, you should disable recursion:

```
    recursion no;
```

If recursion is allowed, it's best to limit this ability to specific clients. This is done with the `allow-recursion-on` and `allow-recursion` directives. The former specifies the network interfaces and the latter the clients that are allowed to perform recursive queries. Two examples:

```
    allow-recursion-on { any; };
    allow-recursion { localnets; };

    allow-recursion-on { eth1; };
    allow-recursion { 192.168.56/24; };
```

36.3.3. forwarders

If you want to configure a forwarder, set the `forward` and `forwarders` directives in the `options` section. The `forward` directive can be `only` (denoting it can only forward queries) or `first` (denoting it will try to resolve the query itself if the forwarder did not respond). The `forwarders` directive specifies the IP addresses of the forwarders. An example with the public Cloudflare DNS servers as forwarders:

```
    forward only;
    forwarders {
        1.1.1.1; 1.0.0.1;
        2606:4700:4700::1111; 2606:4700:4700::1001;
    };
```

36.4. DNS zones

In this section, we'll set up our DNS server to be authoritative for domain `example.com`. We'll create a *forward lookup zone* and a *reverse lookup zone*.

The forward lookup zone will typically contain A, AAAA, CNAME, MX, NS, and SOA records. The reverse lookup zone will contain PTR and SOA records.

A few zone files are already included in the default installation, a.o. the zone file for the *root zone* (see next section), and for the *local domain* (not discussed in this book).

36.4.1. the root hints file

The *root hints file* is a file that contains the addresses of the root servers of the internet. It is used by the DNS server to initiate recursive queries. The file is added during installation, but could -if necessary- be reproduced using the `dig` command. If you call `dig` without arguments, it will query the *dot*, `.` domain, i.e. the root servers. If you send this query to one of the root servers, you will additionally get the glue (A and AAAA) records for each root server:

```
[student@linux ~]$ dig @f.root-servers.net

; <<>> DiG 9.16.23-RH <<>> @f.root-servers.net
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6028
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;.                              IN  NS

;; ANSWER SECTION:
.                       518400  IN  NS    i.root-servers.net.

[ ... more lines of output ... ]

.                       518400  IN  NS    m.root-servers.net.

;; ADDITIONAL SECTION:
i.root-servers.net.  518400  IN  A  192.36.148.17
i.root-servers.net.   518400  IN  AAAA  2001:7fe::53

[ ... more lines of output ... ]

m.root-servers.net.   518400  IN  A     202.12.27.33
m.root-servers.net.   518400  IN  AAAA  2001:dc3::35

;; Query time: 18 msec
;; SERVER: 192.5.5.241#53(192.5.5.241)
;; WHEN: Wed Mar 27 21:25:40 UTC 2024
;; MSG SIZE  rcvd: 811
```

This is exactly the information stored in the root hints file.

36.4.2. forward lookup zone

In order to set up a forward lookup zone, you first need to create a zone file, and then add a zone definition to the main configuration file.

Let's say we want to set up a forward lookup zone for the domain `example.com`. The zone file will have the same name as the zone (`example.com`) and will be stored in the `/var/named` directory (`/etc/bind` on Debian). We want to keep track of the following host names:

Host	Alias	IP	Function
ns1		192.0.2.1	Primary name server
ns2		192.0.2.2	Secondary name server
srv001	www	192.0.2.10	Web server
srv002	mail	192.0.2.20	Mail server

Additionally, we want that `https://example.com/` will also point to the web server.

A zone file for this domain could look like this:

```
;; Zone file for example.com
$ORIGIN example.com.
$TTL 1W

@ IN SOA ns1.example.com. hostmaster.example.com. (
         24061601  ; Serial
         1D        ; Refresh time
         1H        ; Retry time
         1W        ; Expiry time
         1D )      ; Negative cache TTL

; Name servers

       IN  NS     ns1
       IN  NS     ns2

; Mail server

       IN  MX     10 srv002

; Hosts

ns1    IN  A      192.0.2.1
ns2    IN  A      192.0.2.2

srv001 IN  A      192.0.2.10
@      IN  A      192.0.2.10
www    IN  CNAME  srv001

srv002 IN  A      192.0.2.20
smtp   IN  CNAME  srv002
imap   IN  CNAME  srv002
```

The `$ORIGIN` directive on line 1 sets the default domain name for the zone. Fully qualified domain names *must always* end with a dot. Names names *that do not end with a dot* are considered to be relative to this domain, and the value of `$ORIGIN` will be added. E.g. `ns1` will be interpreted as `ns1.example.com.` in this zone file. This is actually a common source of errors in zone files!

The `$TTL` directive on line 2 sets the default *time-to-live* value for the records in the zone. It determines how long a record can be cached by a resolver before it expires. The value `1W` stands for one week.

Line 4 and 5 define the *Start of Authority* record. The `@` symbol is a shorthand for the zone name (`example.com.`). The `IN` keyword stands for *Internet* and is the class of the record. The `SOA` record contains information about the zone:

- `ns.example.com.` is the primary name server for the zone.

- `hostmaster.example.com.` is the email address of the person responsible for the zone (to be interpreted as `hostmaster@example.com`, but the `@` is replaced with a dot because of the special meaning of the `@` symbol in the zone file).
- `24061601` is a serial number that is chosen by the system administrator. It is an integer, but commonly it contains an encoded timestamp, e.g. `YYMMDDHH`. The serial number is used to determine whether a zone transfer is necessary. If the serial number of the primary server is higher than the serial number of the secondary server, a zone transfer will occur. That means that you need to increment the serial number every time you make a change to the zone file.
- `1D` (one day) is the refresh time. It is the time that a secondary server waits before checking if the serial number of the primary server has changed. If it has, it requests a zone transfer.
- `1H` (one hour) is the retry time. It is the time that a secondary server waits before retrying a zone transfer if the previous attempt failed.
- `1W` (one week) is the expiry time. It is the time that a secondary server will keep the zone data if it can't contact the primary server. After this time has elapsed, the secondary server will stop answering queries for the zone.
- `1D` determines how long a NAME ERROR result can be cached.

The `NS` records on line 7 and 8 define the name servers for the zone, i.e. `ns1` and `ns2`.

On line 10, the `MX` record defines the mail server for the domain, i.e. `srv002`.

The `A` records on the following lines define the IP addresses of each host. The first "column" is the (unqualified) hostname and the last is the IP address. Remark that, because the names do not end with a dot, the value of `$ORIGIN` is appended to the names. The record on line 16 with the `@` symbol ensures that an A query for `example.com` points to the web server.

The `CNAME` records, finally, define aliases for the hosts. The `www` alias points to the web server `srv001`, and the `smtp` and `imap` aliases point to the mail server `srv002`

Save the file and test the syntax:

```
[student@el ~]$ sudo vi /var/named/example.com
... edit the file ...
[student@el ~]$ sudo named-checkzone example.com /var/named/example.com
zone example.com/IN: loaded serial 24061601
OK
```

Next, add a zone definition to the main configuration file. The zone definition could look like this:

```
zone "example.com" IN {
  type primary;
  file "example.com";
};
```

The first line defines the domain name of the zone. The `IN` keyword stands for *Internet* and is the class of the zone.

The `type` of this zone is `primary`, meaning that this server is the primary authoritative server for the zone.

The `file` directive specifies the location of the zone file, relative to the directory specified in the `directory` directive in the `options` section (`/var/named` on EL).

Check the syntax, restart the service and test:

```
[student@el ~]$ sudo named-checkconf
[student@el ~]$ sudo systemctl restart named
[student@el ~]$ dig @localhost example.com +short
192.0.2.10
```

Try to query the SOA record from the zone. This is a rare case where `nslookup` gives more information (specifically, the names of the timer fields) than `dig`:

```
[vagrant@el ~]$ dig @localhost SOA example.com +short
ns.example.com. hostmaster.example.com. 24061601 86400 3600 604800 86400
[vagrant@el ~]$ nslookup
> server localhost
Default server: localhost
Address: ::1#53
Default server: localhost
Address: 127.0.0.1#53
> set type=SOA
> example.com
Server:         localhost
Address:        ::1#53

example.com
        origin = ns.example.com
        mail addr = hostmaster.example.com
        serial = 24061601
        refresh = 86400
        retry = 3600
        expire = 604800
        minimum = 86400
```

36.4.3. reverse lookup zone

The example in the previous section is not sufficient to allow the DNS server to respond to *reverse lookup* queries, where the client provides the IP address and wants to know the associated host name. These are specified in a *reverse lookup zone*.

Some domains have IP addresses over several IP subnets. In this case, you will need to create a separate reverse lookup zone for each subnet!

The name of a reverse lookup zone has a special format. For the `example.com` domain, we used the 192.0.2.0/24 IP range. The reverse lookup zone name is constructed as follows:

- Start with the IP addres for the address range: 192.0.2.0/24
- Drop the host part, so only the network part remains: 192.0.2
- Reverse the order of the octets: 2.0.192
- Append `.in-addr.arpa.`: **2.0.192.in-addr.arpa.**

The zone file for this reverse lookup zone could look like this:

```
;; Zone file for reverse lookup zone 2.0.192.in-addr.arpa.
$ORIGIN 2.0.192.in-addr.arpa.
$TTL 1W

@ IN SOA ns1.example.com. hostmaster.example.com. (
         24061601  ; Serial
         1D        ; Refresh time
         1H        ; Retry time
         1W        ; Expiry time
         1D )      ; Negative cache TTL

; Name servers

       IN  NS     ns1.example.com.
```

```
       IN  NS     ns2.example.com.

; Reverse lookup records

1    IN  PTR    ns1.example.com.
2    IN  PTR    ns2.example.com.
10   IN  PTR    srv001.example.com.
20   IN  PTR    srv002.example.com.
```

In this file, we find the SOA record like in the forward lookup zone file. Next, the `NS` records define the name servers for the zone. Finally, the PTR records map an IP address to a host name.

Remark that for the IP addresses, we only need to specify the host part, in this case the last octet. The network part is already defined in the zone name.

Also remark that all host names are fully qualified and end with a dot. If we would only have specified the host name (e.g. `srv001`), the value of `$ORIGIN` would have been appended, resulting in the nonsensical `srv001.2.0.192.in-addr.arpa.`, which is not what we want!

Saving the zone definition to the appropriate file, and check its syntax:

```
[student@el ~]$ sudo vi /var/named/2.0.192.in-addr.arpa
... edit the file ...
[student@el ~]$ sudo named-checkzone 2.0.192.in-addr.arpa /var/named/2.0.192.in-
addr.arpa
zone 2.0.192.in-addr.arpa/IN: loaded serial 24061601
OK
```

Next, we add a zone definition to the main configuration file. The zone definition could look like this:

```
zone "2.0.192.in-addr.arpa" IN {
  type master;
  file "2.0.192.in-addr.arpa";
};
```

At this time, you set up a DNS server that is authoritative for the `example.com` domain, and can respond to forward and reverse lookup queries. If you want to follow best practices, turn off recursion and any forwarders that you might have set up previously.

After adding this section to the main configuration file, check the syntax, restart the service and test:

```
[student@el ~]$ sudo named-checkconf
[student@el ~]$ sudo systemctl restart named
[student@el ~]$ dig @localhost -x 192.0.2.10 +short
srv001.example.com.
```

36.5. secondary server and zone transfer

In this section, we'll set up a secondary server for the `example.com` domain. The secondary server will replicate the zone data from the primary server through a *zone transfer*.

Depending on expected network traffic and server load, a system administrator may want to set up multiple secondary name servers. Usually, the primary server sends notifications to all

Figure 36.3.: Zone transfer from a primary to a secondary name server

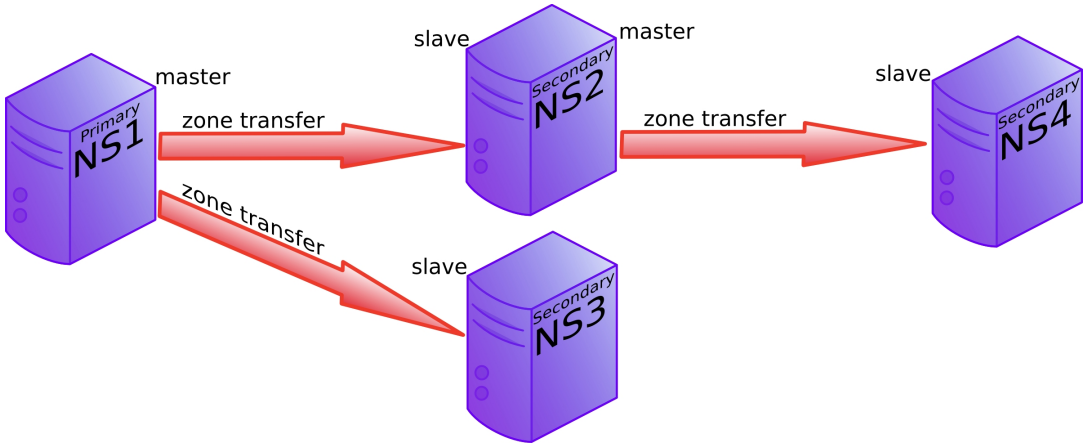


Figure 36.4.: More elaborate setup of primary and secondary name servers. `ns1` notifies `ns2` and `ns3`, but `ns4` is notified by `ns2`.

secondary servers, but sometimes a secondary server can be the primary server for another secondary server.

Zone transfers are requested by the secondary servers at regular intervals. Those intervals are defined in the *SOA record*.

- Set up a new VM (we'll give it host name `el2`), install BIND and start the service, as shown above.
- Ensure the service is listening on all network interfaces and is available for other hosts on the network. Ensure recursion is turned off.

```
[student@el2 ~]$ sudo dnf install -y bind
... output omitted ...
[student@el2 ~]$ sudo systemctl enable --now named
Created symlink /etc/systemd/system/multi-user.target.wants/named.service → /usr/lib/syste
[student@el2 ~]$ sudo vi /etc/named.conf
... edit the file ...
[student@el2 ~]$ sudo named-checkconf
[student@el2 ~]$ sudo systemctl restart named
```

Before we can set up `el2` as a secondary server, we need to allow zone transfers from the primary server. This is done with the `allow-transfer` directive in the zone definition. Add the IP address of the secondary server to the list of allowed hosts:

```
// Zone definitions on theprimary server

zone "example.com" IN {
  type primary;
```

```
  file "example.com";
  notify yes;
  allow-transfer { 192.168.56.111; };
};

zone "2.0.192.in-addr.arpa" IN {
  type primary;
  file "2.0.192.in-addr.arpa";
  notify yes;
  allow-transfer { 192.168.56.111; };
};
```

The `notify` directive tells the server to notify the secondary servers when the zone has changed (or, rather, when the zone's serial has increased).

The `allow-update` directive specifies which hosts are allowed to update the zone.

Remark that the IP address here is the one given to the VM `el2`. It does not correspond with the IP addresses in the zone file, but that is actually not necessary.

Save the file (on the primary server), check the syntax, and restart the service. Ensure query logging is turned on so you can observe the zone transfer in the logs. Follow the logs in real time:

```
[vagrant@el ~]$ sudo vi /etc/named.conf
[vagrant@el ~]$ sudo named-checkconf
[vagrant@el ~]$ sudo systemctl restart named
[vagrant@el ~]$ sudo rndc querylog on
[vagrant@el ~]$ sudo journalctl -flu named
```

Next, set up the secondary server:

```
// Zone definitions on the secondary server

zone "example.com" IN {
    type secondary;
    primaries { 192.168.56.11; };
    file "slaves/example.com";
};

zone "2.0.192.in-addr.arpa" IN {
    type secondary;
    primaries { 192.168.56.11; };
    file "slaves/2.0.192.in-addr.arpa";
};
```

Add these zone definitions for the forward and reverse lookup zones to the main configuration file. Change the IP address to the one of your primary server, if it differs. The secondary server will store the zone database in a file in binary format. On EL, the appropriate directory for these zone files is `/var/named/slaves`, on Debian it is `/var/cache/named`.

Check the syntax and restart. Observe the zone transfer in the primary server logs! Or, additionaly, you can set up a network sniffer to capture the zone transfer.

```
[student@el2 ~]$ sudo vi /etc/named.conf
[student@el2 ~]$ sudo named-checkconf
[student@el2 ~]$ sudo systemctl restart named
[student@el2 ~]$ dig @localhost example.com +short
192.0.2.10
```

The logs on the primary server should show something like this (for clarity, timestamps and other redundent information was removed):

```
query: 2.0.192.in-addr.arpa IN SOA -E(0) (192.168.56.11)
query: 2.0.192.in-addr.arpa IN AXFR -T (192.168.56.11)
transfer of '2.0.192.in-addr.arpa/IN': AXFR started (serial 24061601)
transfer of '2.0.192.in-addr.arpa/IN': AXFR ended: 1 messages, 8 records, 248 bytes, 0.001 s
query: example.com IN SOA -E(0) (192.168.56.11)
query: example.com IN AXFR -T (192.168.56.11)
transfer of 'example.com/IN': AXFR started (serial 24061601)
transfer of 'example.com/IN': AXFR ended: 1 messages, 13 records, 317 bytes, 0.001 secs (317
```

	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.37	192.168.1.35	DNS	Standard query SOA cobbaut.paul
2	0.008502	192.168.1.35	192.168.1.37	DNS	Standard query response SOA ns.cobbaut.paul
3	0.014672	192.168.1.37	192.168.1.35	TCP	33713 > domain [SYN] Seq=0 Win=5840 Len=0 MS
4	0.015215	192.168.1.35	192.168.1.37	TCP	domain > 33713 [SYN, ACK] Seq=0 Ack=1 Win=57
5	0.015307	192.168.1.37	192.168.1.35	TCP	33713 > domain [ACK] Seq=1 Ack=1 Win=5856 Le
6	0.015954	192.168.1.37	192.168.1.35	TCP	[TCP segment of a reassembled PDU]
7	0.018359	192.168.1.35	192.168.1.37	TCP	domain > 33713 [ACK] Seq=1 Ack=3 Win=5792 Le
8	0.018411	192.168.1.37	192.168.1.35	DNS	Standard query IXFR cobbaut.paul
9	0.018823	192.168.1.35	192.168.1.37	TCP	domain > 33713 [ACK] Seq=1 Ack=77 Win=5792 L
10	0.019784	192.168.1.35	192.168.1.37	DNS	Standard query response SOA ns.cobbaut.paul
11	0.019821	192.168.1.37	192.168.1.35	TCP	33713 > domain [ACK] Seq=77 Ack=295 Win=6912
12	0.020618	192.168.1.37	192.168.1.35	TCP	33713 > domain [FIN, ACK] Seq=77 Ack=295 Win
13	0.021011	192.168.1.35	192.168.1.37	TCP	domain > 33713 [FIN, ACK] Seq=295 Ack=78 Win
14	0.021040	192.168.1.37	192.168.1.35	TCP	33713 > domain [ACK] Seq=78 Ack=296 Win=6912

Figure 36.5.: A zone transfer captured by Wireshark.

The transfer was performed using an AXFR query, requesting a *full zone transfer*. You can run this query yourself from the command line on the secondary server:

```
[student@el2 ~]$ dig @192.168.56.11 AXFR example.com

; <<>> DiG 9.16.23-RH <<>> @192.168.56.11 AXFR example.com
; (1 server found)
;; global options: +cmd
example.com.        604800  IN    SOA    ns.example.com. hostmaster.example.com. 24061601 8
example.com.            604800  IN      A       192.0.2.10
example.com.            604800  IN      NS      ns1.example.com.
example.com.            604800  IN      NS      ns2.example.com.
example.com.            604800  IN      MX      10 srv002.example.com.
imap.example.com.       604800  IN      CNAME   srv002.example.com.
ns1.example.com.        604800  IN      A       192.0.2.1
ns2.example.com.        604800  IN      A       192.0.2.2
smtp.example.com.       604800  IN      CNAME   srv002.example.com.
srv001.example.com.     604800  IN      A       192.0.2.10
srv002.example.com.     604800  IN      A       192.0.2.20
www.example.com.        604800  IN      CNAME   srv001.example.com.
example.com.        604800  IN    SOA    ns.example.com. hostmaster.example.com. 24061601 8
;; Query time: 2 msec
;; SERVER: 192.168.56.11#53(192.168.56.11)
;; WHEN: Sun Jun 16 09:21:32 UTC 2024
;; XFR size: 13 records (messages 1, bytes 356)
```

This returns all the records in the zone file for the `example.com` domain. We hope that this also illustrates the security risk of allowing zone transfers to any host! The AXFR query is quite

useful for an attacker who is trying to enumerate all the hosts in a domain. So be careful with this and only allow zone transfers to secondary name servers.

You can also force a refresh from a zone with `rndc`. The example below forces a transfer of the `example.com` zone:

```
[student@el2 ~]$ sudo rndc retransfer example.com
```

There also exists an *incremental zone transfer* (IXFR), which only transfers the changes since the last transfer. The decision on which of the two (AXFR/IXFR) depends on the size of the transfer that is needed to completely update the zone on the secondary server. An incremental zone transfer is prefered when the total size of changes is smaller than the size of the zone database.

36.6. practice: BIND

Use Vagrant and VirtualBox to set up the following scenario for DNS domain `linuxtrn.lan` with IP range 172.16.76.0/24. The scenario is illustrated in the following diagram:

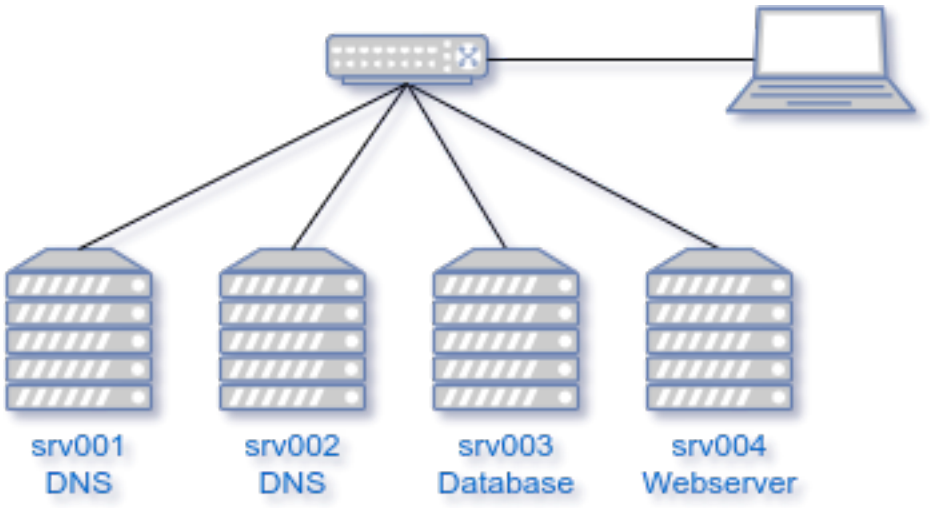


Figure 36.6.: BIND practice lab

Four VMs are attached to a common Host-Only or Internal network. The properties of the VMs are summarized in the following table:

Host	Alias	IP	Role
`srv001`	`ns1`	`172.16.76.251`	Primary DNS server
`srv002`	`ns2`	`172.16.76.252`	Secondary DNS server
`srv003`	`smtp`, `imap`	`172.16.76.3`	Mail server
`srv004`	`www`	`172.16.76.4`	Webserver

If you used a Host-Only network, the host machine can play the role of a client. In the case of an internal network, add another VM (e.g. a ready-made Linux Mint or Kali Linux VM) to the network as a client. In this scenario, routing is not considered. The VMs will have internet access through their NAT interfaces.

Remark that the VMS `srv003` and `srv004` should not necessarily exist in order to make te setup work. It could add to the realism of the scenario, though, since you can check whether entering `https://www.linuxtrn.lan` in a browser on a client will work.

1. Set up `srv001` (Debian or EL-based) and install BIND.

   - Discover the default configuration files. Can you define the purpose of each file?
   - Turn on the query log and start `tcpdump` to capture any traffic for port 53 (inbound and outbound).
   - Without changing the configuration, send a simple forward A query for any domain name from the VM itself. Do you expect this to work?
   - Try to determine from the logs or the `tcpdump` output whether the DNS server is configured as a *caching name server* with or without a *forwarder*.
   - Repeat the previous query. Do you see a difference in behavior? Can you explain why?

2. Ensure that the DNS server is available to all hosts on the local network. Don't forget to check firewall settings! Test whether you can resolve the domain name from another VM or a physical machine on the same network.

3. Add a *forwarder* and verify that it works. Try to use a public DNS server as a forwarder. Google's is well known, but there are others, too! Search for "free public dns servers" to get some suggestions.

   - Repeat the queries from the previous step. Do you see a difference in behavior?

4. Create a *primary forward lookup zone* named `linuxtrn.lan` with a variety of resource records, e.g. NS, MX, A, CNAME. Also add an A record for the `@` shorthand.

   - Use `dig` and `nslookup` to verify all resource records.
   - Optionally, write a test script that runs these queries automatically and compares the results with the expected values.

5. On the client machine, set the system DNS server to the IP address of `srv001` and test (from the terminal, or using your test script) whether you can:

   - Ping the VMs by their hostnames.
   - Resolve the domain name `linuxtrn.lan` to an IP address.
   - Query the name and mail servers for the domain `linuxtrn.lan`.
   - Perform a reverse lookup for any of the IP addresses in the domain.
   - Access the website on `srv004` by entering `https://www.linuxtrn.lan` in a browser (or alternatively use `curl` if your client VM does not have a graphical UI).

6. Set up `srv002`, install BIND and set up a *secondary* server for your primary zone.

   - Rewrite your test script so it can also run queries against the secondary server.
   - Ensure that you have the query logs on the primary server turned on and that you are watching its logs before you start the secondary server.
   - Observe the zone transfer process when you start the secondary server.
   - Check that the secondary server responds to the same queries as the primary.
   - Try an AXFR query from the secondary server to the primary server. Try the same from the client machine. Also try it from the primary server to the secondary server. Which work and which don't?
   - Make a change to one of the resource records on the primary server (e.g. change an IP address) and update your test script. Restart the primary server. Query both the primary and the secondary server to see if the change has been propagated.
   - It probably hasn't, unless you thought of incrementing the seral number in the zone file. Do that now and repeat the previous step. Check whether the zone transfer happens and that both primary and secondary server respond with the updated information.

36.7. solution: BIND

TODO: update to the new version of the exercises

1. Set up a Linux VM (Debian or EL-based) and install BIND. Verify with a sniffer how it works.

   You should see queries to the root name servers with tcpdump or wireshark.

2. Add a *forwarder* and verify that it works.

   The forwarder van be added in named.conf.options as seen in the theory.

3. Create a *primary forward lookup zone* named `yourname.local` with at least two NS records and four A records.

   This is literally explained in the theory.

4. Use `dig` and `nslookup` to verify your NS and A records.

   This is literally explained in the theory.

5. Create a new VM, install BIND and set up a *secondary* server of your primary zone. Verify the *zone transfer* in the logs.

   This is literally explained in the theory.

6. Set up two primary zones on two servers and implement a *conditional forwarder* (you can use the two servers from before).

   A conditional forwarder is set in `named.conf.local` as a zone. (see the theory on forwarder)

Part XII.

RAID; expert installation

37. introduction to raid

(Written by Paul Cobbaut, https://github.com/paulcobbaut/)

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

Where ten years ago nobody was arguing about the best choice being hardware RAID, this has changed since technologies like mdadm, lvm and even zfs focus more on managability. The workload on the cpu for software RAID used to be high, but cpu's have gotten a lot faster.

37.1. raid levels

37.1.1. raid 0

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

37.1.2. jbod

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

37.1.3. raid 1

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

37.1.4. raid 2, 3 and 4 ?

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

37.1.5. raid 5

`raid 5` uses `three` or more disks, each divided into chunks. Every time chunks are written to the array, one of the disks will receive a `parity` chunk. Unlike `raid 4`, the parity chunk will alternate between all disks. The main advantage of this is that `raid 5` will allow for full data recovery in case of `one` hard disk failure.

37.1.6. raid 6

`raid 6` is very similar to `raid 5`, but uses two parity chunks. `raid 6` protects against two hard disk failures. Oracle Solaris `zfs` calls this `raidz2` (and also had `raidz3` with triple parity).

37.1.7. raid 0+1

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

37.1.8. raid 1+0

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

37.1.9. raid 50

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

37.1.10. many others

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

37.2. building a software raid5 array

37.2.1. do we have three disks?

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

```
[root@linux ~]# fdisk -l 2> /dev/null | grep MB
Disk /dev/sdb: 8589 MB, 8589934592 bytes
Disk /dev/sdc: 8589 MB, 8589934592 bytes
Disk /dev/sdd: 8589 MB, 8589934592 bytes
```

37.2.2. fd partition type

The next step is to create a partition of type `fd` on every disk. The `fd` type is to set the partition as `Linux RAID autodetect`. See this (truncated) screenshot:

```
[root@linux ~]# fdisk /dev/sdd
...
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1044, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-1044, default 1044):
Using default value 1044

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

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

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

37.2.3. verify all three partitions

Now all three disks are ready for `raid 5`, so we have to tell the system what to do with these disks.

```
[root@linux ~]# fdisk -l 2> /dev/null | grep raid
/dev/sdb1       1        1044     8385898+  fd  Linux raid autodetect
/dev/sdc1       1        1044     8385898+  fd  Linux raid autodetect
/dev/sdd1       1        1044     8385898+  fd  Linux raid autodetect
```

37.2.4. create the raid5

The next step used to be *create the* `raid table` *in* `/etc/raidtab`. Nowadays, you can just issue the command `mdadm` with the correct parameters.

The command below is split on two lines to fit this print, but you should type it on one line, without the backslash (\).

```
[root@linux ~]# mdadm --create /dev/md0 --chunk=64 --level=5 --raid-\
devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Below a partial screenshot how fdisk -l sees the `raid 5`.

```
[root@linux ~]# fdisk -l /dev/md0

Disk /dev/md0: 17.2 GB, 17172135936 bytes
2 heads, 4 sectors/track, 4192416 cylinders
Units = cylinders of 8 * 512 = 4096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 65536 bytes / 131072 bytes
Disk identifier: 0×00000000

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

We could use this software `raid 5` array in the next topic: `lvm`.

37.2.5. /proc/mdstat

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

```
[root@linux ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdd1[3] sdc1[1] sdb1[0]
      16769664 blocks super 1.2 level 5, 64k chunk, algorithm 2 [3/2] [UU_]
      [============>........]  recovery = 62.8% (5266176/8384832) finish=0\
.3min speed=139200K/sec
```

This example shows an active software `raid 5`.

```
[root@linux ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdd1[3] sdc1[1] sdb1[0]
    16769664 blocks super 1.2 level 5, 64k chunk, algorithm 2 [3/3] [UUU]
```

37.2.6. mdadm --detail

Use `mdadm --detail` to get information on a raid device.

```
[root@linux ~]# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sun Jul 17 13:48:41 2011
     Raid Level : raid5
     Array Size : 16769664 (15.99 GiB 17.17 GB)
  Used Dev Size : 8384832 (8.00 GiB 8.59 GB)
   Raid Devices : 3
  Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Sun Jul 17 13:49:43 2011
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 0
  Spare Devices : 0
```

```
     Layout : left-symmetric
 Chunk Size : 64K

       Name : rhel6c:0  (local to host rhel6c)
       UUID : c10fd9c3:08f9a25f:be913027:999c8e1f
     Events : 18

Number   Major   Minor   RaidDevice State
   0       8       17        0      active sync   /dev/sdb1
   1       8       33        1      active sync   /dev/sdc1
   3       8       49        2      active sync   /dev/sdd1
```

37.2.7. removing a software raid

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

```
[root@linux ~]# mdadm --stop /dev/md0
mdadm: stopped /dev/md0
```

The disks can now be repartitioned.

37.2.8. further reading

Take a look at the man page of `mdadm` for more information. Below an example command to add a new partition while removing a faulty one.

```
mdadm /dev/md0 --add /dev/sdd1 --fail /dev/sdb1 --remove /dev/sdb1
```

37.3. practice: raid

1. Add three virtual disks of 1GB each to a virtual machine.
2. Create a software `raid 5` on the three disks. (It is not necessary to put a filesystem on it)
3. Verify with `fdisk` and in `/proc` that the `raid 5` exists.
4. Stop and remove the `raid 5`.
5. Create a `raid 1` to mirror two disks.

37.4. solution: raid

1. Add three virtual disks of 1GB each to a virtual machine.
2. Create a software `raid 5` on the three disks. (It is not necessary to put a filesystem on it)
3. Verify with `fdisk` and in `/proc` that the `raid 5` exists.
4. Stop and remove the `raid 5`.
5. Create a `raid 1` to mirror two disks.

```
[root@linux ~]# mdadm --create /dev/md0 --level=1 --raid-devices=2 \
/dev/sdb1 /dev/sdc1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
[root@linux ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1]
md0 : active raid1 sdc1[1] sdb1[0]
      8384862 blocks super 1.2 [2/2] [UU]
      [====>.............]  resync = 20.8% (1745152/8384862) \
finish=0.5min speed=218144K/sec
```

A. git

(Written by Paul Cobbaut, https://github.com/paulcobbaut/, with contributions by: Alex M. Schapelle, https://github.com/zero-pytagoras/)

This chapter is an introduction to using `git` on the command line. The `git repository` is hosted by `github`, but you are free to choose another server (or create your own).

There are many excellent online tutorials for `git`. This list can save you one Google query:

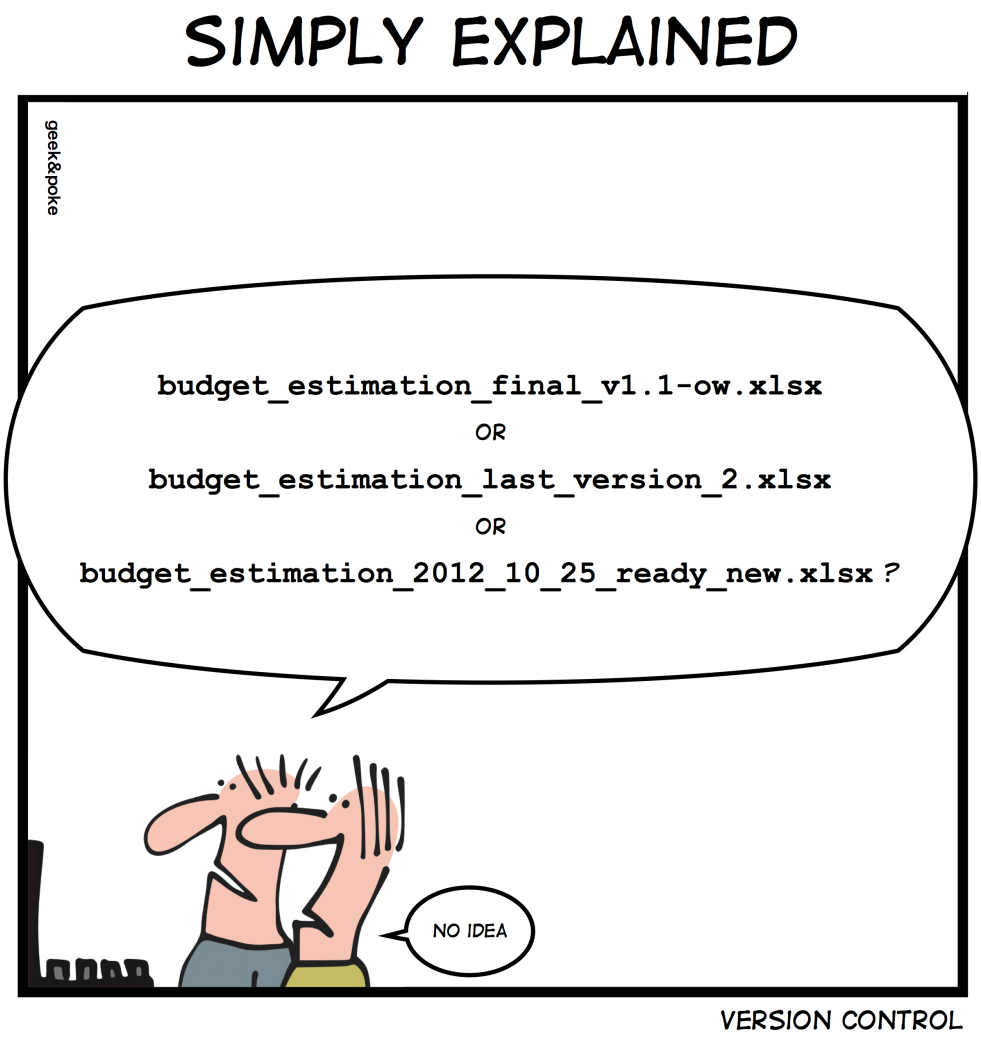
```
http://gitimmersion.com/
http://git-scm.com/book
```

A.1. git

Linus Torvalds created `git` back in 2005 when Bitkeeper changed its license and the Linux kernel developers where no longer able to use it for free.

`git` quickly became popular and is now the most widely used `distributed version control` system in the world.

Geek and Poke demonstrates why we need version control (image property of Geek and Poke CCA 3.0).



Besides `source code` for software, you can also find German and Icelandic `law` on github (and probably much more by the time you are reading this).

A.2. installing git

We install `git` with `aptitude install git` as seen in this screenshot on Debian 6.

```
root@linux:~# aptitude install git
The following NEW packages will be installed:
  git libcurl3-gnutls{a} liberror-perl{a}
0 packages upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
...
Processing triggers for man-db ...
Setting up libcurl3-gnutls (7.21.0-2.1+squeeze2) ...
Setting up liberror-perl (0.17-1) ...
Setting up git (1:1.7.2.5-3) ...
```

A.3. starting a project

First we create a project directory, with a simple file in it.

```
student@linux~$ mkdir project42
student@linux~$ cd project42/
student@linux~/project42$ echo "echo The answer is 42." >> question.sh
```

A.3.1. git init

Then we tell `git` to create an empty git repository in this directory.

```
student@linux~/project42$ ls -la
total 12
drwxrwxr-x  2 paul paul 4096 Dec  8 16:41 .
drwxr-xr-x 46 paul paul 4096 Dec  8 16:41 ..
-rw-rw-r--  1 paul paul   23 Dec  8 16:41 question.sh
student@linux~/project42$ git init
Initialized empty Git repository in /home/paul/project42/.git/
student@linux~/project42$ ls -la
total 16
drwxrwxr-x  3 paul paul 4096 Dec  8 16:44 .
drwxr-xr-x 46 paul paul 4096 Dec  8 16:41 ..
drwxrwxr-x  7 paul paul 4096 Dec  8 16:44 .git
-rw-rw-r--  1 paul paul   23 Dec  8 16:41 question.sh
```

A.3.2. git config

Next we use `git config` to set some global options.

```
student@linux$ git config --global user.name Paul
student@linux$ git config --global user.email "paul.cobbaut@gmail.com"
student@linux$ git config --global core.editor vi
```

We can verify this config in `~/.gitconfig`:

```
student@linux~/project42$ cat ~/.gitconfig
[user]
    name = Paul
    email = paul.cobbaut@gmail.com
[core]
    editor = vi
```

A.3.3. git add

Time now to add file to our project with `git add`, and verify that it is added with `git status`.

```
student@linux~/project42$ git add question.sh
student@linux~/project42$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   question.sh
#
```

The `git status` tells us there is a new file ready to be committed.

A.3.4. git commit

With `git commit` you force git to record all added files (and all changes to those files) permanently.

```
student@linux~/project42$ git commit -m "starting a project"
[master (root-commit) 5c10768] starting a project
 1 file changed, 1 insertion(+)
 create mode 100644 question.sh
student@linux~/project42$ git status
# On branch master
nothing to commit (working directory clean)
```

A.3.5. changing a committed file

The screenshots below show several steps. First we change a file:

```
student@linux~/project42$ git status
# On branch master
nothing to commit (working directory clean)
student@linux~/project42$ vi question.sh
```

Then we verify the status and see that it is modified:

```
student@linux~/project42$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   question.sh
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Next we add it to the git repository.

```
student@linux~/project42$ git add question.sh
student@linux~/project42$ git commit -m "adding a she-bang to the main script"
[master 86b8347] adding a she-bang to the main script
 1 file changed, 1 insertion(+)
student@linux~/project42$ git status
# On branch master
nothing to commit (working directory clean)
```

A.3.6. git log

We can see all our commits again using `git log`.

```
student@linux~/project42$ git log
commit 86b8347192ea025815df7a8e628d99474b41fb6c
Author: Paul <paul.cobbaut@gmail.com>
Date:   Sat Dec 8 17:12:24 2012 +0100

    adding a she-bang to the main script

commit 5c10768f29aecc16161fb197765e0f14383f7bca
Author: Paul <paul.cobbaut@gmail.com>
Date:   Sat Dec 8 17:09:29 2012 +0100

    starting a project
```

The log format can be changed.

```
student@linux~/project42$ git log --pretty=oneline
86b8347192ea025815df7a8e628d99474b41fb6c adding a she-bang to the main script
5c10768f29aecc16161fb197765e0f14383f7bca starting a project
```

The log format can be customized a lot.

```
student@linux~/project42$ git log --pretty=format:"%an: %ar :%s"
Paul: 8 minutes ago :adding a she-bang to the main script
Paul: 11 minutes ago :starting a project
```

A.3.7. git mv

Renaming a file can be done with `mv` followed by a `git remove` and a `git add` of the new filename. But it can be done easier and in one command using `git mv`.

```
student@linux~/project42$ git mv question.sh thequestion.sh
student@linux~/project42$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   renamed:    question.sh -> thequestion.sh
#
student@linux~/project42$ git commit -m "improved naming scheme"
[master 69b2c8b] improved naming scheme
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename question.sh => thequestion.sh (100%)
```

A.4. git branches

Working on the project can be done in one or more `git branches`. Here we create a new branch that will make changes to the script. We will `merge` this branch with the `master branch` when we are sure the script works. (It can be useful to add `git status` commands when practicing).

```
student@linux~/project42$ git branch
* master
student@linux~/project42$ git checkout -b newheader
Switched to a new branch 'newheader'
student@linux~/project42$ vi thequestion.sh
student@linux~/project42$ git add thequestion.sh
student@linux~/project42$ source thequestion.sh
The answer is 42.
```

It seems to work, so we commit in this branch.

```
student@linux~/project42$ git commit -m "adding a new company header"
[newheader 730a22b] adding a new company header
 1 file changed, 4 insertions(+)
student@linux~/project42$ git branch
  master
* newheader
student@linux~/project42$ cat thequestion.sh
#!/bin/bash
#
# copyright linux-training.be
#

echo The answer is 42.
```

Let us go back to the master branch and see what happened there.

```
student@linux~/project42$ git checkout master
Switched to branch 'master'
student@linux~/project42$ cat thequestion.sh
#!/bin/bash
echo The answer is 42.
```

Nothing happened in the master branch, because we worked in another branch.

When we are sure the branch is ready for production, then we merge it into the master branch.

```
student@linux~/project42$ cat thequestion.sh
#!/bin/bash
echo The answer is 42.
student@linux~/project42$ git merge newheader
Updating 69b2c8b..730a22b
Fast-forward
 thequestion.sh |    4 ++++
 1 file changed, 4 insertions(+)
student@linux~/project42$ cat thequestion.sh
#!/bin/bash
#
# copyright linux-training.be
#

echo The answer is 42.
```

The newheader branch can now be deleted.

```
student@linux~/project42$ git branch
* master
  newheader
student@linux~/project42$ git branch -d newheader
Deleted branch newheader (was 730a22b).
student@linux~/project42$ git branch
* master
```

A.5. to be continued...

The `git` story is not finished.

There are many excellent online tutorials for `git`. This list can save you one Google query:

```
http://gitimmersion.com/
http://git-scm.com/book
```

A.6. github.com

Create an account on `github.com`. This website is a frontend for an immense git server with over two and a half million users and almost five million projects (including Fedora, Linux kernel, Android, Ruby on Rails, Wine, X.org, VLC...)

```
https://github.com/signup/free
```

This account is free of charge, we will use it in the examples below.

A.7. add your public key to github

I prefer to use github with a `public key`, so it probably is a good idea that you also upload your public key to github.com.

You can upload your own key via the web interface:

```
https://github.com/settings/ssh
```

Please do not forget to protect your `private key`!

A.8. practice: git

1.Crate local project called `git_practice`.

2.Create a project on `gitlab.com` to host a local project that you have created.

3.The project should have `REAMDE.md` file as well as `TODO.md` file in it.

4.Write in `REAMDE.md` file description of the project and what you think it might be.

5.Initialize your project with `git` command, setup your username, mail and remote server.

6.Use `git push -u origin master` to send project saves to remote host.

7.Verify on `gitlab.com` that the project has been setup and is updated with `REAMDE.md` and `TODO.md`.

8.Add git_hello.sh script that prints hello to username from its current location.

9.Push the script to gitlab repository.

A.9. solution: git

1.Crate local project called `git_practice`.

```
aschapelle@vaio3:~$ mkdir git_practice; cd git_practice
```

2.Create a project on `gitlab.com` to host a local project that you have created.

3.The project should have `REAMDE.md` file as well as `TODO.md` file in it.

```
aschapelle@vaio3:~/git_practice$ touch REAMDE.md TODO.md
```

4.Write in `REAMDE.md` file description of the project and what you think it might be.

```
aschapelle@vaio3:~/git_practice$ echo "This is readme file for git_practice project" > READM
        aschapelle@vaio3:~/git_practice$echo "This is todo file for git_practice project" >
```

5.Initialize your project with `git` command, setup your username, mail and remote server.

```
aschapelle@vaio3:~/git_practice$ git init
        aschapelle@vaio3:~/git_practice$ git config user.name alex.schapelle
      aschapelle@vaio3:~/git_practice$ git config user.mail alex@vaiolabs.com
     aschapelle@vaio3:~/git_practice$ git remote add origin https://gitlab.com/url_to_you
```

6.Use `git push -u origin master` to send project saves to remote host.

```
        aschapelle@vaio3:~/git_practice$ git push -u origin master
```

7.Verify on `gitlab.com` that the project has been setup and is updated with `REAMDE.md` and `TODO.md`.

8.Add git_hello.sh script that prints hello to username from its current location.

```
aschapelle@vaio3:~/git_practice$ git push -u origin master
```

9.Push the script to gitlab repository.

```
        aschapelle@vaio3:~/git_practice$ git push -u origin master
```

B. Introduction to vi

(Written by Paul Cobbaut, https://github.com/paulcobbaut/)

The `vi` editor is installed on almost every Unix. Linux will very often install `vim` (`vi improved`) which is similar. Every system administrator should know `vi(m)`, because it is an easy tool to solve problems.

The `vi` editor is not intuitive, but once you get to know it, `vi` becomes a very powerful application. Most Linux distributions will include the `vimtutor` which is a 45 minute lesson in `vi(m)`.

B.1. command mode and insert mode

The vi editor starts in `command mode`. In command mode, you can type commands. Some commands will bring you into `insert mode`. In insert mode, you can type text. The `escape key` will return you to command mode.

Table B.1.: getting to command mode

key	action
Esc	set vi(m) in command mode.

B.2. start typing (a A i I o O)

The difference between a A i I o and O is the location where you can start typing. a will append after the current character and A will append at the end of the line. i will insert before the current character and I will insert at the beginning of the line. o will put you in a new line after the current line and O will put you in a new line before the current line.

Table B.2.: switch to insert mode

command	action
a	start typing after the current character
A	start typing at the end of the current line
i	start typing before the current character
I	start typing at the start of the current line
o	start typing on a new line after the current line
O	start typing on a new line before the current line

B.3. replace and delete a character (r x X)

When in command mode (it doesn't hurt to hit the escape key more than once) you can use the x key to delete the current character. The big X key (or shift x) will delete the character left of the cursor. Also when in command mode, you can use the r key to replace one single character. The r key will bring you in insert mode for just one key press, and will return you immediately to command mode.

Table B.3.: replace and delete

command	action
x	delete the character below the cursor
X	delete the character before the cursor
r	replace the character below the cursor
p	paste after the cursor (here the last deleted character)
xp	switch two characters

B.4. undo, redo and repeat (u .)

When in command mode, you can undo your mistakes with u. Use `ctrl-r` to redo the undo.

You can do your mistakes twice with . (in other words, the . will repeat your last command).

Table B.4.: undo and repeat

command	action
u	undo the last action
ctrl-r	redo the last undo
.	repeat the last action

B.5. cut, copy and paste a line (dd yy p P)

When in command mode, dd will cut the current line. yy will copy the current line. You can paste the last copied or cut line after (p) or before (P) the current line.

Table B.5.: cut, copy and paste a line

command	action
dd	cut the current line
yy	(yank yank) copy the current line
p	paste after the current line
P	paste before the current line

B.6. cut, copy and paste lines (3dd 2yy)

When in command mode, before typing dd or yy, you can type a number to repeat the command a number of times. Thus, 5dd will cut 5 lines and 4yy will copy (yank) 4 lines. That last one will be noted by vi in the bottom left corner as "4 line yanked".

Table B.6.: cut, copy and paste lines

command	action
3dd	cut three lines
4yy	copy four lines

B.7. start and end of a line (0 or ^ and $)

When in command mode, the 0 and the caret ^ will bring you to the start of the current line, whereas the $ will put the cursor at the end of the current line. You can add 0 and $ to the d command, d0 will delete every character between the current character and the start of the line. Likewise d$ will delete everything from the current character till the end of the line. Similarly y0 and y$ will yank till start and end of the current line.

Table B.7.: start and end of line

command	action
0	jump to start of current line
^	jump to start of current line
$	jump to end of current line
d0	delete until start of line
d$	delete until end of line

B.8. join two lines (J) and more

When in command mode, pressing `J` will append the next line to the current line. With `yyp` you duplicate a line and with `ddp` you switch two lines.

Table B.8.: join two lines

command	action
J	join two lines
yyp	duplicate a line
ddp	switch two lines

B.9. words (w b)

When in command mode, `w` will jump to the next word and `b` will move to the previous word. w and b can also be combined with d and y to copy and cut words (dw db yw yb).

Table B.9.: words

command	action
w	forward one word
b	back one word
3w	forward three words
dw	delete one word
yw	yank (copy) one word

command	action
5yb	yank five words back
7dw	delete seven words

B.10. save (or not) and exit (:w :q :q!)

Pressing the colon : will allow you to give instructions to vi (technically speaking, typing the colon will open the `ex` editor). `:w` will write (save) the file, `:q` will quit an unchanged file without saving, and `:q!` will quit vi discarding any changes. `:wq` will save and quit and is the same as typing `ZZ` in command mode.

Table B.10.: save and exit vi

command	action
:w	save (write)
:w fname	save as fname
:q	quit
:wq	save and quit
ZZ	save and quit
:q!	quit (discarding your changes)
:w!	save (and write to non-writable file!)

The last one is a bit special. With `:w! vi` will try to `chmod` the file to get write permission (this works when you are the owner) and will `chmod` it back when the write succeeds. This should always work when you are root (and the file system is writable).

B.11. Searching (/ ?)

When in command mode typing / will allow you to search in vi for strings (can be a regular expression). Typing /foo will do a forward search for the string foo and typing ?bar will do a backward search for bar.

Table B.11.: searching

command	action
/string	forward search for string
?string	backward search for string
n	go to next occurrence of search string
/^string	forward search string at beginning of line
/string$	forward search string at end of line
/br[aeio]l	search for bral brel bril and brol
/\<he\>	search for the word `he` (and not for `he`re or t`he`)

B.12. replace all (:1,$ s/foo/bar/g)

To replace all occurrences of the string foo with bar, first switch to ex mode with : . Then tell vi which lines to use, for example 1,$ will do the replace all from the first to the last line. You can write 1,5 to only process the first five lines. The s/foo/bar/g will replace all occurrences of foo with bar.

Table B.12.: replace

command	action
:4,8 s/foo/bar/g	replace foo with bar on lines 4 to 8
:1,$ s/foo/bar/g	replace foo with bar on all lines

B.13. reading files (:r :r !cmd)

When in command mode, :r foo will read the file named foo, :r !foo will execute the command foo. The result will be put at the current location. Thus :r !ls will put a listing of the current directory in your text file.

Table B.13.: read files and input

command	action
:r fname	(read) file fname and paste contents
:r !cmd	execute cmd and paste its output

B.14. text buffers

There are 36 buffers in vi to store text. You can use them with the " character.

Table B.14.: text buffers

command	action
"add	delete current line and put text in buffer a
"g7yy	copy seven lines into buffer g
"ap	paste from buffer a

B.15. multiple files

You can edit multiple files with vi. Here are some tips.

Table B.15.: multiple files

command	action
vi file1 file2 file3	start editing three files
:args	lists files and marks active file
:n	start editing the next file
:e	toggle with last edited file
:rew	rewind file pointer to first file

B.16. abbreviations

With `:ab` you can put abbreviations in vi. Use `:una` to undo the abbreviation.

Table B.16.: abbreviations

command	action
:ab str long string	abbreviate `str` to be 'long string'
:una str	un-abbreviate str

B.17. key mappings

Similarly to their abbreviations, you can use mappings with `:map` for command mode and `:map!` for insert mode.

This example shows how to set the F6 function key to toggle between `set number` and `set nonumber`. The <bar> separates the two commands, `set number!` toggles the state and `set number?` reports the current state.

```
:map <F6> :set number!<bar>set number?<CR>
```

B.18. setting options

Some options that you can set in vim.

```
:set number  ( also try :se nu )
:set nonumber
:syntax on
:syntax off
:set all  (list all options)
:set tabstop=8
:set tx   (CR/LF style endings)
:set notx
```

You can set these options (and much more) in `~/.vimrc` for vim or in `~/.exrc` for standard vi.

```
student@linux:~$ cat ~/.vimrc
set number
set tabstop=8
set textwidth=78
map <F6> :set number!<bar>set number?<CR>
student@linux:~$
```

B.19. practice: vi(m)

1. Start the vimtutor and do some or all of the exercises. You might need to run `aptitude install vim` on xubuntu.

2. What 3 key sequence in command mode will duplicate the current line.

3. What 3 key sequence in command mode will switch two lines' place (line five becomes line six and line six becomes line five).

4. What 2 key sequence in command mode will switch a character's place with the next one.

5. vi can understand macro's. A macro can be recorded with q followed by the name of the macro. So qa will record the macro named a. Pressing q again will end the recording. You can recall the macro with @ followed by the name of the macro. Try this example: i 1 'Escape Key' qa yyp 'Ctrl a' q 5@a (Ctrl a will increase the number with one).

6. Copy /etc/passwd to your ~/passwd. Open the last one in vi and press Ctrl v. Use the arrow keys to select a Visual Block, you can copy this with y or delete it with d. Try pasting it.

7. What does dwwP do when you are at the beginning of a word in a sentence ?

B.20. solution: vi(m)

1. Start the vimtutor and do some or all of the exercises. You might need to run `aptitude install vim` on xubuntu.

```
vimtutor
```

2. What 3 key sequence in command mode will duplicate the current line.

```
yyp
```

3. What 3 key sequence in command mode will switch two lines' place (line five becomes line six and line six becomes line five).

```
ddp
```

4. What 2 key sequence in command mode will switch a character's place with the next one.

```
xp
```

5. vi can understand macro's. A macro can be recorded with q followed by the name of the macro. So qa will record the macro named a. Pressing q again will end the recording. You can recall the macro with @ followed by the name of the macro. Try this example: i 1 'Escape Key' qa yyp 'Ctrl a' q 5@a (Ctrl a will increase the number with one).

6. Copy /etc/passwd to your ~/passwd. Open the last one in vi and press Ctrl v. Use the arrow keys to select a Visual Block, you can copy this with y or delete it with d. Try pasting it.

```
cp /etc/passwd ~
vi passwd
(press Ctrl-V)
```

7. What does `dwwP` do when you are at the beginning of a word in a sentence ?

`dwwP` can switch the current word with the next word.

C. GNU Free Documentation License

Version 1.3, 3 November 2008

C.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

C.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this

License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

C.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

C.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

C.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

C.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

C.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

C.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

C.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

C.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

C.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

C.12. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently

incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.