Chapter

# 8

# Basic Configuration: Network, Accounts, Printing...

Contents



*A computer with a new installation created with `debian-installer` is intended to be as functional as possible, but many services still have to be configured. Furthermore, it is always good to know how to change certain configuration elements defined during the initial installation process.*

This chapter reviews everything included in what we could call the "basic configuration": networking, language and locales, users and groups, printing, mount points, etc.

## 8.1. Configuring the System for Another Language

If the system was installed using French, the machine will probably already have French set as the default language. But it is good to know what the installer does to set the language, so that later, if the need arises, you can change it.

TOOL

**The `locale` command to display the current configuration**

The `locale` command lists a summary of the current configuration of various locale parameters (date format, numbers format, etc.), presented in the form of a group of standard environment variables dedicated to the dynamic modification of these settings.

### 8.1.1. Setting the Default Language

A locale is a group of regional settings. This includes not only the language for text, but also the format for displaying numbers, dates, times, and monetary sums, as well as the alphabetical comparison rules (to properly account for accented characters). Although each of these parameters can be specified independently from the others, we generally use a locale, which is a coherent set of values for these parameters corresponding to a "region" in the broadest sense. These locales are usually indicated under the form, *language-code_COUNTRY-CODE*, sometimes with a suffix to specify the character set and encoding to be used. This enables consideration of idiomatic or typographical differences between different regions with a common language.

CULTURE

**Character sets**

Historically, each locale has an associated "character set" (group of known characters) and a preferred "encoding" (internal representation for characters within the computer).

The most popular encodings for latin-based languages were limited to 256 characters because they opted to use a single byte for each character. Since 256 characters was not enough to cover all European languages, multiple encodings were needed, and that is how we ended up with *ISO-8859-1* (also known as "Latin 1") up to *ISO-8859-15* (also known as "Latin 9"), among others.

Working with foreign languages often implied regular switches between various encodings and character sets. Furthermore, writing multilingual documents led to further, almost intractable problems. Unicode (a super-catalog of nearly all writing systems from all of the world's languages) was created to work around this problem. One of Unicode's encodings, UTF-8, retains all 128 ASCII symbols (7-bit codes), but handles other characters differently. Those are preceded by a specific escape sequence of a few bits, which implicitly defines the length of the character. This allows encoding all Unicode characters on a sequence of one or more bytes. Its use has been popularized by the fact that it is the default encoding in XML documents.

This is the encoding that should generally be used, and is thus the default on Debian systems.

The *locales* package includes all the elements required for proper functioning of "localization" of various applications. During installation, this package will ask to select a set of supported languages. This set can be changed at any time by running `dpkg-reconfigure locales` as root.

The first question invites you to select "locales" to support. Selecting all English locales (meaning those beginning with "en_") is a reasonable choice. Do not hesitate to also enable other locales if the machine will host foreign users. The list of locales enabled on the system is stored in the `/etc/locale.gen` file. It is possible to edit this file by hand, but you should run `locale-gen` after any modifications. It will generate the necessary files for the added locales to work, and remove any obsolete files.

The second question, entitled "Default locale for the system environment", requests a default locale. The recommended choice in the U.S.A. is "en_US.UTF-8". British English speakers will prefer "en_GB.UTF-8", and Canadians will prefer either "en_CA.UTF-8" or, for French, "fr_CA.UTF-8". The `/etc/default/locale` file will then be modified to store this choice. From there, it is picked up by all user sessions since PAM will inject its content in the `LANG` environment variable.

The *locales-all* package contains the precompiled locale data for all supported locales.

BEHIND THE SCENES

**`/etc/environment` and `/etc/default/locale`**

The `/etc/environment` file provides the `login`, `gdm`, or even `ssh` programs with the correct environment variables to be created.

These applications do not create these variables directly, but rather via a PAM (`pam_env.so`) module. PAM (Pluggable Authentication Module) is a modular library centralizing the mechanisms for authentication, session initialization, and password management. See section 11.7.3.2, "Configuring PAM" page 315 for an example of PAM configuration.

The `/etc/default/locale` file works in a similar manner, but contains only the `LANG` environment variable. Thanks to this split, some PAM users can inherit a complete environment without localization. Indeed, it is generally discouraged to run server programs with localization enabled; on the other hand, localization and regional settings are recommended for programs that open user sessions.

### 8.1.2. Configuring the Keyboard

Even if the keyboard layout is managed differently in console and graphical mode, Debian offers a single configuration interface that works for both: it is based on debconf and is implemented in the *keyboard-configuration* package. Thus the `dpkg-reconfigure keyboard-configuration` command can be used at any time to reset the keyboard layout.

The questions are relevant to the physical keyboard layout (a standard PC keyboard in the US will be a "Generic 104 key"), then the layout to choose (generally "US"), and then the position of the AltGr key (right Alt). Finally comes the question of the key to use for the "Compose key", which allows for entering special characters by combining keystrokes. Type successively Compose ' e and produce an e-acute ("é"). All these combinations are described in the `/usr/`

`share/X11/locale/en_US.UTF-8/Compose` file (or another file, determined according to the current locale indicated by `/usr/share/X11/locale/compose.dir`).

Note that the keyboard configuration for graphical mode described here only affects the default layout; the GNOME and KDE Plasma environments, among others, provide a keyboard control panel in their preferences allowing each user to have their own configuration. Some additional options regarding the behavior of some particular keys are also available in these control panels.

### 8.1.3. Migrating to UTF-8

The generalization of UTF-8 encoding has been a long awaited solution to numerous difficulties with interoperability, since it facilitates international exchange and removes the arbitrary limits on characters that can be used in a document. The one drawback is that it had to go through a rather difficult transition phase. Since it could not be completely transparent (that is, it could not happen at the same time all over the world), two conversion operations were required: one on file contents, and the other on filenames. Fortunately, the bulk of this migration has been completed and we discuss it largely for reference.

CULTURE

***Mojibake* and interpretation errors**

When a text is sent (or stored) without encoding information, it is not always possible for the recipient to know with certainty what convention to use for determining the meaning of a set of bytes. You can usually get an idea by getting statistics on the distribution of values present in the text, but that doesn't always give a definite answer. When the encoding system chosen for reading differs from that used in writing the file, the bytes are mis-interpreted, and you get, at best, errors on some characters, or, at worst, something completely illegible.

Thus, if a French text appears normal with the exception of accented letters and certain symbols which appear to be replaced with sequences of characters like "Ã©" or Ã¨" or "Ã§", it is probably a file encoded as UTF-8 but interpreted as ISO-8859-1 or ISO-8859-15. This is a sign of a local installation that has not yet been migrated to UTF-8. If, instead, you see question marks instead of accented letters — even if these question marks seem to also replace a character that should have followed the accented letter — it is likely that your installation is already configured for UTF-8 and that you have been sent a document encoded in Western ISO.

So much for "simple" cases. These cases only appear in Western culture, since Unicode (and UTF-8) was designed to maximize the common points with historical encodings for Western languages based on the Latin alphabet, which allows recognition of parts of the text even when some characters are missing.

In more complex configurations, which, for example, involve two environments corresponding to two different languages that do not use the same alphabet, you often get completely illegible results — a series of abstract symbols that have nothing to do with each other. This is especially common with Asian languages due to their numerous languages and writing systems. The Japanese word *mojibake* has been adopted to describe this phenomenon. When it appears, diagnosis is more complex and the simplest solution is often to simply migrate to UTF-8 on both sides.

As far as file names are concerned, the migration can be relatively simple. The `convmv` tool (in the package with the same name) was created specifically for this purpose; it allows renaming

files from one encoding to another. The use of this tool is relatively simple, but we recommend doing it in two steps to avoid surprises. The following example illustrates a UTF-8 environment containing directory names encoded in ISO-8859-15, and the use of `convmv` to rename them.

```
$ ls travail/
Ic?nes  ?l?ments graphiques  Textes
$ convmv -r -f iso-8859-15 -t utf-8 travail/
Starting a dry run without changes...
mv "travail/�l�ments graphiques"        "travail/Éléments graphiques"
mv "travail/Ic�nes"     "travail/Icônes"
No changes to your files done. Use --notest to finally rename the files.
$ convmv -r --notest -f iso-8859-15 -t utf-8 travail/
mv "travail/�l�ments graphiques"        "travail/Éléments graphiques"
mv "travail/Ic�nes"     "travail/Icônes"
Ready!
$ ls travail/
Éléments graphiques  Icônes  Textes
```

For the file content, conversion procedures are more complex due to the vast variety of existing file formats. Some file formats include encoding information that facilitates the tasks of the software used to treat them; it is sufficient, then, to open these files and re-save them specifying UTF-8 encoding. In other cases, you have to specify the original encoding (ISO-8859-1 or "Western", or ISO-8859-15 or "Western (Euro)", according to the formulations) when opening the file.

For simple text files, you can use `recode` (in the package of the same name) which allows automatic recoding. This tool has numerous options so you can play with its behavior. We recommend you consult the documentation, the `recode(1)` man page, or the `recode` info page (more complete).

## 8.2. Configuring the Network

BACK TO BASICS

**Essential network concepts (Ethernet, IP address, subnet, broadcast)**

Most modern local networks use the Ethernet protocol, where data is split into small blocks called frames and transmitted on the wire one frame at a time. Data speeds vary from 10 Mb/s for older Ethernet cards to 100 Gb/s in the newest cards (with the most common rate currently growing from 100 Mb/s to 10 Gb/s). The most widely used cables are called 10BASE-T, 100BASE-T, 1000BASE-T, 10GBASE-T and 40GBASE-T, depending on the throughput they can reliably provide (the T stands for "twisted pair"); those cables end in an RJ45 connector. There are other cable types, used mostly for speeds of 10 Gb/s and above.

An IP address is a number used to identify a network interface on a computer on a local network or the Internet. In the currently most widespread version of IP (IPv4), this number is encoded in 32 bits, and is usually represented as 4 numbers separated by periods (e.g. `192.168.0.1`), each number being between 0 and 255 (inclusive, which corresponds to 8 bits of data). The next version of the protocol, IPv6, extends this addressing space to 128 bits, and the addresses are generally represented as a series of hexadecimal numbers separated by colons (e.g., 2001:0db8:13bb:0002:0000:0000:0000:0020, or 2001:db8:13bb:2::20 for short).

A subnet mask (netmask) defines in its binary code which portion of an IP address corresponds to the network, the remainder specifying the machine. In the example of configuring a static IPv4 address given here, the subnet mask, `255.255.255.0` (24 "1"s followed by 8 "0"s in binary representation) indicates that the first 24 bits of the IP address correspond to the network address, and the other 8 are specific to the machine. In IPv6, for readability, only the number of "1"s is expressed; the netmask for an IPv6 network could, thus, be `64`.

The network address is an IP address in which the part describing the machine's number is `0`. The range of IPv4 addresses in a complete network is often indicated by the syntax, *a.b.c.d/e*, in which *a.b.c.d* is the network address and *e* is the number of bits affected to the network part in an IP address. The example network would thus be written: `192.168.0.0/24`. The syntax is similar in IPv6: `2001:db8:13bb:2::/64`.

A router is a machine that connects several networks to each other. All traffic coming through a router is guided to the correct network. To do this, the router analyzes incoming packets and redirects them according to the IP address of their destination. The router is often known as a gateway; in this configuration, it works as a machine that helps reach out beyond a local network (towards an extended network, such as the Internet).

The special broadcast address connects all the stations in a network. Almost never "routed", it only functions on the network in question. Specifically, it means that a data packet addressed to the broadcast never passes through the router.

This chapter focuses on IPv4 addresses, since they are currently the most commonly used. The details of the IPv6 protocol are approached in section 10.6, "IPv6" page 257, but the concepts remain the same.

The network is automatically configured during the initial installation. If Network Manager gets installed (which is generally the case for full desktop installations), then it might be that no configuration is actually required (for example, if you rely on DHCP on a wired connection and have no specific requirements). If a configuration is required (for example, for a WiFi interface), then it will create the appropriate file in `/etc/NetworkManager/system-connections/`.

NOTE

**NetworkManager**

If Network Manager is particularly recommended in roaming setups (see section 8.2.5, "Automatic Network Configuration for Roaming Users" page 169), it is also perfectly usable as the default network management tool. You can create "System connections" that are used as soon as the computer boots either manually with a `.ini`-like file in `/etc/NetworkManager/system-connections/` or through a graphical tool (`nm-connection-editor`). If you were using ifupdown, just remember to deactivate the entries in `/etc/network/interfaces` that you want Network Manager to handle.

➨ `https://wiki.gnome.org/Projects/NetworkManager/SystemSettings`

➨ `https://developer.gnome.org/NetworkManager/1.14/ref-settings.html`

If Network Manager is not installed, then the installer will configure *ifupdown* by creating the `/etc/network/interfaces` file. A line starting with `auto` gives a list of interfaces to be auto-

matically configured on boot by the networking service. When there are many interfaces, it is good practice to keep the configuration in different files inside `/etc/network/interfaces.d/`.

In a server context, *ifupdown* is thus the network configuration tool that you usually get. That is why we will cover it in the next sections.

### 8.2.1. Ethernet Interface

If the computer has an Ethernet card, the IP network that is associated with it must be configured by choosing from one of two methods. The simplest method is dynamic configuration with DHCP, and it requires a DHCP server on the local network. It may indicate a desired hostname, corresponding to the hostname setting in the example below. The DHCP server then sends configuration settings for the appropriate network.

**Example 8.1** *DHCP configuration*

```
auto enp0s31f6
iface enp0s31f6 inet dhcp
  hostname arrakis
```

IN PRACTICE

**Names of network interfaces**

By default, the kernel attributes generic names such a `eth0` (for wired Ethernet) or `wlan0` (for WiFi) to the network interfaces. The number in those names is a simple incremental counter representing the order in which they have been detected. With modern hardware, that order might change for each reboot and thus the default names are not reliable.

Fortunately, systemd and udev are able to rename the interfaces as soon as they appear. The default name policy is defined by `/lib/systemd/network/99-default.link` (see `systemd.link(5)` for an explanation of the `NamePolicy` entry in that file). In practice, the names are often based on the device's physical location (as guessed by where they are connected) and you will see names starting with `en` for wired ethernet and `wl` for WiFi. In the example above, the rest of the name indicates, in abbreviated form, a PCI (`p`) bus number (`0`), a slot number (`s31`), a function number (`f6`).

Obviously, you are free to override this policy and/or to complement it to customize the names of some specific interfaces. You can find out the names of the network interfaces in the output of `ip addr` (or as filenames in `/sys/class/net/`).

In some corner cases it might be necessary to disable the consistent naming of network devices as described above. Besides changing the default *udev* rule it is also possible to boot the system using the `net.ifnames=0` and `biosdevname=0` kernel parameters to achieve that.

A "static" configuration must indicate network settings in a fixed manner. This includes at least the IP address and subnet mask; network and broadcast addresses are also sometimes listed. A router connecting to the exterior will be specified as a gateway.

**Example 8.2** *Static configuration*

```
auto enp0s31f6
iface enp0s31f6 inet static
  address 192.168.0.3/24
  broadcast 192.168.0.255
  network 192.168.0.0
  gateway 192.168.0.1
```

NOTE

**Multiple addresses**

It is possible not only to associate several interfaces to a single, physical network card, but also several IP addresses to a single interface. Remember also that an IP address may correspond to any number of names via DNS, and that a name may also correspond to any number of numerical IP addresses.

As you can guess, the configurations can be rather complex, but these options are only used in very special cases. The examples cited here are typical of the usual configurations.

### 8.2.2. Wireless Interface

Getting wireless network cards to work can be a bit more challenging. First of all, they often require the installation of proprietary firmwares which are not installed by default in Debian. Then wireless networks rely on cryptography to restrict access to authorized users only, this implies storing some secret key in the network configuration. Let's tackle those topics one by one.

#### *Installing the required firmwares*

First you have to enable the non-free repository in APT's sources.list file: see section 6.1, "Filling in the `sources.list` File" page 108 for details about this file. Many firmware are proprietary and are thus located in this repository. You can try to skip this step if you want, but if the next step doesn't find the required firmware, retry after having enabled the non-free section.

Then you have to install the appropriate firmware-* packages. If you don't know which package you need, you can install the *isenkram* package and run its `isenkram-autoinstall-firmware` command. The packages are often named after the hardware manufacturer or the corresponding kernel module: *firmware-iwlwifi* for Intel wireless cards, *firmware-atheros* for Qualcomm Atheros, *firmware-ralink* for Ralink, etc. A reboot is then recommended because the kernel driver usually looks for the firmware files when it is first loaded and no longer afterwards.

*Wireless specific entries in `/etc/network/interfaces`*

*ifupdown* is able to manage wireless interfaces but it needs the help of the *wpasupplicant* package which provides the required integration between *ifupdown* and the `wpa_supplicant` command used to configure the wireless interfaces (when using WPA/WPA2 encryption). The usual entry in `/etc/network/interfaces` needs to be extended with two supplementary parameters to specify the name of the wireless network (aka its SSID) and the *Pre-Shared Key* (PSK).

**Example 8.3** *DHCP configuration for a wireless interface*

```
auto wlp4s0
iface wlp4s0 inet dhcp
  wpa-ssid Falcot
  wpa-psk ccb290fd4fe6b22935cbae31449e050edd02ad44627b16ce0151668f5f53c01b
```

The wpa-psk parameter can contain either the plain text passphrase or its hashed version generated with `wpa_passphrase *SSID passphrase*`. If you use an unencrypted wireless connection, then you should put a wpa-key-mgmt NONE and no wpa-psk entry. For more information about the possible configuration options, have a look at `/usr/share/doc/wpasupplicant/README.Debian.gz`.

At this point, you should consider restricting the read permissions on `/etc/network/interfaces` to the root user only since the file contains a private key that not all users should have access to.

HISTORY

**WEP encryption**

Usage of the deprecated WEP encryption protocol is possible with the *wireless-tools* package. See `/usr/share/doc/wireless-tools/README.Debian` for instructions.

### 8.2.3. Connecting with PPP through a PSTN Modem

A point to point (PPP) connection establishes an intermittent connection; this is the most common solution for connections made with a telephone modem ("PSTN modem", since the connection goes over the public switched telephone network).

A connection by telephone modem requires an account with an access provider, including a telephone number, username, password, and, sometimes the authentication protocol to be used. Such a connection is configured using the `pppconfig` tool in the Debian package of the same name. By default, it sets up a connection named provider (as in Internet service provider). When in doubt about the authentication protocol, choose *PAP*: it is offered by the majority of Internet service providers.

After configuration, it is possible to connect using the `pon` command (giving it the name of the connection as a parameter, when the default value of provider is not appropriate). The link is

disconnected with the `poff` command. These two commands can be executed by the root user, or by any other user, provided they are in the `dip` group.

### 8.2.4. Connecting through an ADSL Modem

The generic term "ADSL modem" covers a multitude of devices with very different functions. The modems that are simplest to use with Linux are those that have an Ethernet interface (and not only a USB interface). These tend to be popular; most ADSL Internet service providers lend (or lease) a "box" with Ethernet interfaces. Depending on the type of modem, the configuration required can vary widely.

#### *Modems Supporting PPPOE*

Some Ethernet modems work with the PPPOE protocol (Point to Point Protocol over Ethernet). The `pppoeconf` tool (from the package with the same name) will configure the connection. To do so, it modifies the `/etc/ppp/peers/dsl-provider` file with the settings provided and records the login information in the `/etc/ppp/pap-secrets` and `/etc/ppp/chap-secrets` files. It is recommended to accept all modifications that it proposes.

Once this configuration is complete, you can open the ADSL connection with the command, `pon dsl-provider` and disconnect with `poff dsl-provider`.

TIP

**Starting ppp at boot**

PPP connections over ADSL are, by definition, intermittent. Since they are usually not billed according to time, there are few downsides to the temptation of keeping them always open. The standard means to do so is to use the init system.

With systemd, adding an automatically restarting task for the ADSL connection is a simple matter of creating a "unit file" such as `/etc/systemd/system/adsl-connection.service`, with contents such as the following:

```
[Unit]
Description=ADSL connection

[Service]
Type=forking
ExecStart=/usr/sbin/pppd call dsl-provider
Restart=always

[Install]
WantedBy=multi-user.target
```

Once this unit file has been defined, it needs to be enabled with `systemctl enable adsl-connection`. Then the loop can be started manually with `systemctl start adsl-connection`; it will also be started automatically on boot.

On systems not using `systemd` (including *Wheezy* and earlier versions of Debian), the standard SystemV init works differently. On such systems, all that is needed is to add a line such as the following at the end of the `/etc/inittab` file; then, any time the connection is disconnected, `init` will reconnect it.

```
adsl:2345:respawn:/usr/sbin/pppd call dsl-provider
```

For ADSL connections that auto-disconnect on a daily basis, this method reduces the duration of the interruption.

### *Modems Supporting PPTP*

The PPTP (Point-to-Point Tunneling Protocol) protocol was created by Microsoft. Deployed at the beginning of ADSL, it was quickly replaced by PPPOE. If this protocol is forced on you, see section 10.3.4, "PPTP" page 250.

### *Modems Supporting DHCP*

When a modem is connected to the computer by an Ethernet cable (crossover cable) you typically configure a network connection by DHCP on the computer; the modem automatically acts as a gateway by default and takes care of routing (meaning that it manages the network traffic between the computer and the Internet).

BACK TO BASICS

**Crossover cable for a direct Ethernet connection**

Computer network cards expect to receive data on specific wires in the cable, and send their data on others. When you connect a computer to a local network, you usually connect a cable (straight or crossover) between the network card and a repeater or switch. However, if you want to connect two computers directly (without an intermediary switch or repeater), you must route the signal sent by one card to the receiving side of the other card, and vice-versa. This is the purpose of a crossover cable, and the reason it is used.

Note that this distinction has become almost irrelevant over time, as modern network cards are able to detect the type of cable present and adapt accordingly, so it won't be unusual that both kinds of cable will work in a given location.

Most "ADSL routers" on the market can be used like this, as do most of the ADSL modems provided by Internet services providers.

## 8.2.5. Automatic Network Configuration for Roaming Users

Many Falcot engineers have a laptop computer that, for professional purposes, they also use at home. The network configuration to use differs according to location. At home, it may be a wifi network (protected by a WPA key), while the workplace uses a wired network for greater security and more bandwidth.

To avoid having to manually connect or disconnect the corresponding network interfaces, administrators installed the *network-manager* package on these roaming machines. This software enables a user to easily switch from one network to another using a small icon displayed in the

notification area of their graphical desktop. Clicking on this icon displays a list of available networks (both wired and wireless), so they can simply choose the network they wish to use. The program saves the configuration for the networks to which the user has already connected, and automatically switches to the best available network when the current connection drops.

In order to do this, the program is structured in two parts: a daemon running as root handles activation and configuration of network interfaces and a user interface controls this daemon. PolicyKit handles the required authorizations to control this program and Debian configured PolicyKit in such a way so that members of the netdev group can add or change Network Manager connections.

Network Manager knows how to handle various types of connections (DHCP, manual configuration, local network), but only if the configuration is set with the program itself. This is why it will systematically ignore all network interfaces in `/etc/network/interfaces` and `/etc/network/interfaces.d/` for which it is not suited. Since Network Manager doesn't give details when no network connections are shown, the easy way is to delete from `/etc/network/interfaces` any configuration for all interfaces that must be managed by Network Manager.

Note that this program is installed by default when the "Desktop Environment" task is chosen during initial installation.

## 8.3. Setting the Hostname and Configuring the Name Service

The purpose of assigning names to IP numbers is to make them easier for people to remember. In reality, an IP address identifies a network interface associated with a device such as a network card. Since each machine can have several network cards, and several interfaces on each card, one single computer can have several names in the domain name system.

Each machine is, however, identified by a main (or "canonical") name, stored in the `/etc/hostname` file and communicated to the Linux kernel by initialization scripts through the `hostname` command. The current value is available in a virtual filesystem, and you can get it with the `cat /proc/sys/kernel/hostname` command.

BACK TO BASICS

**`/proc/` and `/sys/`, virtual filesystems**

The `/proc/` and `/sys/` file trees are generated by "virtual" filesystems. This is a practical means of recovering information from the kernel (by listing virtual files) and communicating them to it (by writing to virtual files).

`/sys/` in particular is designed to provide access to internal kernel objects, especially those representing the various devices in the system. The kernel can, thus, share various pieces of information: the status of each device (for example, if it is in energy saving mode), whether it is a removable device, etc. Note that `/sys/` has only existed since kernel version 2.6. `/proc/` describes the current state of the kernel: the files in this directory contain information about the processes running on the system and its hardware.

Surprisingly, the domain name is not managed in the same way, but comes from the complete name of the machine, acquired through name resolution. You can change it in the `/etc/hosts`

file; simply write a complete name for the machine there at the beginning of the list of names associated with the address of the machine, as in the following example:

```
127.0.0.1     localhost
192.168.0.1   arrakis.falcot.com arrakis
```

### 8.3.1. Name Resolution

The mechanism for name resolution in Linux is modular and can use various sources of information declared in the `/etc/nsswitch.conf` file. The entry that involves host name resolution is hosts. By default, it contains files dns, which means that the system consults the `/etc/hosts` file first, then DNS servers. NIS/NIS+ or LDAP servers are other possible sources.

NOTE

**NSS and DNS**

Be aware that the commands specifically intended to query DNS (especially `host`) do not use the standard name resolution mechanism (NSS). As a consequence, they do not take into consideration `/etc/nsswitch.conf`, and thus, not `/etc/hosts` either.

#### *Configuring DNS Servers*

DNS (Domain Name Service) is a distributed and hierarchical service mapping names to IP addresses, and vice-versa. Specifically, it can turn a human-friendly name such as www.eyrolles.com into the actual IP address, 213.244.11.247.

To access DNS information, a DNS server must be available to relay requests. Falcot Corp has its own, but an individual user is more likely to use the DNS servers provided by their ISP.

The DNS servers to be used are indicated in the `/etc/resolv.conf`, one per line, with the nameserver keyword preceding an IP address, as in the following example:

```
nameserver 212.27.32.176
nameserver 212.27.32.177
nameserver 8.8.8.8
```

Note that the `/etc/resolv.conf` file may be handled automatically (and overwritten) when the network is managed by NetworkManager or configured via DHCP.

#### *The `/etc/hosts` file*

If there is no name server on the local network, it is still possible to establish a small table mapping IP addresses and machine hostnames in the `/etc/hosts` file, usually reserved for local network stations. The syntax of this file as described in `hosts(5)` is very simple: each line indicates a specific IP address followed by the list of any associated names (the first being "completely qualified", meaning it includes the domain name).

This file is available even during network outages or when DNS servers are unreachable, but will only really be useful when duplicated on all the machines on the network. The slightest alteration in correspondence will require the file to be updated everywhere. This is why `/etc/hosts` generally only contains the most important entries.

This file will be sufficient for a small network not connected to the Internet, but with 5 machines or more, it is recommended to install a proper DNS server.

TIP

**Bypassing DNS**

Since applications check the `/etc/hosts` file before querying DNS, it is possible to include information in there that is different from what the DNS would return, and therefore to bypass normal DNS-based name resolution.

This allows, in the event of DNS changes not yet propagated, to test access to a website with the intended name even if this name is not properly mapped to the correct IP address yet.

Another possible use is to redirect traffic intended for a specific host to the localhost, thus preventing any communication with the given host. For example, hostnames of servers dedicated to serving ads could be diverted which would bypass these ads resulting in more fluid, less distracting, navigation.

## 8.4. User and Group Databases

The list of users is usually stored in the `/etc/passwd` file, while the `/etc/shadow` file stores hashed passwords. Both are text files, in a relatively simple format, which can be read and modified with a text editor. Each user is listed there on a line with several fields separated with a colon (":").

NOTE

**Editing system files**

The system files mentioned in this chapter are all plain text files, and can be edited with a text editor. Considering their importance to core system functionality, it is always a good idea to take extra precautions when editing system files. First, always make a copy or backup of a system file before opening or altering it. Second, on servers or machines where more than one person could potentially access the same file at the same time, take extra steps to guard against file corruption.

For this purpose, it is enough to use the `vipw` command to edit the `/etc/passwd` file, or `vigr` to edit `/etc/group`. These commands lock the file in question prior to running the text editor, (`vi` by default, unless the `EDITOR` environment variable has been altered). The `-s` option in these commands allows editing the corresponding *shadow* file.

BACK TO BASICS

**Crypt, a one-way function**

`crypt` is a one-way function that transforms a string (`A`) into another string (`B`) in a way that `A` cannot be derived from `B`. The only way to identify `A` is to test all possible values, checking each one to determine if transformation by the function will produce `B` or not. It uses up to 8 characters as input (string `A`) and generates a string of 13, printable, ASCII characters (string `B`).

### 8.4.1. User List: `/etc/passwd`

Here is the list of fields in the `/etc/passwd` file:

- login, for example rhertzog;
- password: this is a password encrypted by a one-way function (`crypt`), relying on DES, MD5, SHA-256 or SHA-512. The special value "x" indicates that the encrypted password is stored in `/etc/shadow`;
- uid: unique number identifying each user;
- gid: unique number for the user's main group (Debian creates a specific group for each user by default);
- GECOS: data field usually containing the user's full name;
- login directory, assigned to the user for storage of their personal files (the environment variable `$HOME` generally points here);
- program to execute upon login. This is usually a command interpreter (shell), giving the user free rein. If you specify `/bin/false` (which does nothing and returns control immediately), the user cannot login.

BACK TO BASICS

**Unix group**

A Unix group is an entity including several users so that they can easily share files using the integrated permission system (by benefiting from the same rights). You can also restrict use of certain programs to a specific group.

### 8.4.2. The Hidden and Encrypted Password File: `/etc/shadow`

The `/etc/shadow` file contains the following fields:

- login;
- encrypted password;
- several fields managing password expiration.

SECURITY

**`/etc/shadow` file security**

`/etc/shadow`, unlike its alter-ego, `/etc/passwd`, cannot be read by regular users. Any hashed password stored in `/etc/passwd` is readable by anybody; a cracker could try to "break" (or reveal) a password by one of several "brute force" methods which, simply put, guess at commonly used combinations of characters. This attack — called a "dictionary attack" — is no longer possible on systems using `/etc/shadow`.

DOCUMENTATION

**`/etc/passwd`, `/etc/shadow` and `/etc/group` file formats**

These formats are documented in the following man pages: `passwd(5)`, `shadow(5)`, and `group(5)`.

### 8.4.3. Modifying an Existing Account or Password

The following commands allow modification of the information stored in specific fields of the user databases: `passwd` permits a regular user to change their password, which in turn, updates the `/etc/shadow` file; `chfn` (CHange Full Name), reserved for the super-user (root), modifies the GECOS field. `chsh` (CHange SHell) allows the user to change their login shell; however, available choices will be limited to those listed in `/etc/shells`; the administrator, on the other hand, is not bound by this restriction and can set the shell to any program of their choosing.

Finally, the `chage` (CHange AGE) command allows the administrator to change the password expiration settings (the `-l` *user* option will list the current settings). You can also force the expiration of a password using the `passwd -e` *user* command, which will require the user to change their password the next time they log in.

### 8.4.4. Disabling an Account

You may find yourself needing to "disable an account" (lock out a user), as a disciplinary measure, for the purposes of an investigation, or simply in the event of a prolonged or definitive absence of a user. A disabled account means the user cannot login or gain access to the machine. The account remains intact on the machine and no files or data are deleted; it is simply inaccessible. This is accomplished by using the command `passwd -l` *user* (lock). Re-enabling the account is done in similar fashion, with the `-u` option (unlock).

GOING FURTHER

**NSS and system databases**

Instead of using the usual files to manage lists of users and groups, you could use other types of databases, such as LDAP or db, by using an appropriate NSS (Name Service Switch) module. The modules used are listed in the `/etc/nsswitch.conf` file, under the `passwd`, `shadow` and `group` entries. See section 11.7.3.1, "Configuring NSS" page 313 for a specific example of the use of an NSS module by LDAP.

### 8.4.5. Group List: `/etc/group`

Groups are listed in the `/etc/group` file, a simple textual database in a format similar to that of the `/etc/passwd` file, with the following fields:

- group name;
- password (optional): This is only used to join a group when one is not a usual member (with the `newgrp` or `sg` commands, see sidebar "Working with several groups" page 175);
- gid: unique group identification number;
- list of members: list of names of users who are members of the group, separated by commas.

BACK TO BASICS

**Working with several groups**

Each user may be a member of many groups; one of them is their "main group". A user's main group is, by default, created during initial user configuration. By default, each file that a user creates belongs to them, as well as to their main group. This is not always desirable; for example, when the user needs to work in a directory shared by a group other than their main group. In this case, the user needs to change their main group using one of the following commands: `newgrp`, which starts a new shell, or `sg`, which simply executes a command using the supplied alternate group. These commands also allow the user to join a group to which they do not belong. If the group is password protected, they will need to supply the appropriate password before the command is executed.

Alternatively, the user can set the `setgid` bit on the directory, which causes files created in that directory to automatically belong to the correct group. For more details, see sidebar "`setgid` directory and *sticky bit*" page 214.

The `id` command displays the current state of a user, with their personal identifier (`uid` variable), current main group (`gid` variable), and the list of groups to which they belong (`groups` variable).

The `addgroup` and `delgroup` commands add or delete a group, respectively. The `groupmod` command modifies a group's information (its gid or identifier). The command `gpasswd` *`group`* changes the password for the group, while the `gpasswd -r` *`group`* command deletes it.

TIP

**`getent`**

The `getent` (get entries) command checks the system databases the standard way, using the appropriate library functions, which in turn call the NSS modules configured in the `/etc/nsswitch.conf` file. The command takes one or two arguments: the name of the database to check, and a possible search key. Thus, the command `getent passwd rhertzog` will give the information from the user database regarding the user `rhertzog`.

## 8.5. Creating Accounts

One of the first actions an administrator needs to do when setting up a new machine is to create user accounts. This is typically done using the `adduser` command which takes a user-name for the new user to be created, as an argument.

The `adduser` command asks a few questions before creating the account, but its usage is fairly straightforward. Its configuration file, `/etc/adduser.conf`, includes all the interesting settings: it can be used to automatically set a quota for each new user by creating a user template, or to change the location of user accounts; the latter is rarely useful, but it comes in handy when you have a large number of users and want to divide their accounts over several disks, for instance. You can also choose a different default shell.

BACK TO BASICS

**Quota**

The term "quota" refers to a limit on machine resources that a user is allowed to use. This frequently refers to disk space.

The creation of an account populates the user's home directory with the contents of the `/etc/skel/` template. This provides the user with a set of standard directories and configuration files.

In some cases, it will be useful to add a user to a group (other than their default "main" group) in order to grant them additional permissions. For example, a user who is included in the *audio* group can access audio devices (see sidebar "Device access permissions" page 176). This can be achieved with a command such as `adduser` *`user group`*.

BACK TO BASICS

**Device access permissions**

Each hardware peripheral device is represented under Unix with a special file, usually stored in the file tree under `/dev/` (DEVices). Two types of special files exist according to the nature of the device: "character mode" and "block mode" files, each mode allowing for only a limited number of operations. While character mode limits interaction with read/write operations, block mode also allows seeking within the available data. Finally, each special file is associated with two numbers ("major" and "minor") that identify the device to the kernel in a unique manner. Such a file, created by the `mknod` command, simply contains a symbolic (and more human-friendly) name.

The permissions of a special file map to the permissions necessary to access the device itself. Thus, a file such as `/dev/mixer`, representing the audio mixer, only has read/write permissions for the root user and members of the `audio` group. Only these users can operate the audio mixer.

It should be noted that the combination of *udev* and *policykit* can add additional permissions to allow users physically connected to the console (and not through the network) to access to certain devices.

## 8.6. Shell Environment

Command interpreters (or shells) can be a user's first point of contact with the computer, and they must therefore be rather friendly. Most of them use initialization scripts that allow configuration of their behavior (automatic completion, prompt text, etc.).

`bash`, the standard shell, uses the `/etc/bash.bashrc` initialization script for "interactive" shells, and `/etc/profile` for "login" shells.

BACK TO BASICS

**Login shell and (non) interactive shell**

In simple terms, a login shell is invoked when you login to the console either locally or remotely via `ssh`, or when you run an explicit `bash --login` command. Regardless of whether it is a login shell or not, a shell can be interactive (in an `xterm`-type terminal for instance); or non-interactive (when executing a script).

DISCOVERY

**Other shells, other scripts**

Each command interpreter has a specific syntax and its own configuration files. Thus, `zsh` uses `/etc/zshrc` and `/etc/zshenv`; `tcsh` uses `/etc/csh.cshrc`, `/etc/csh.login` and `/etc/csh.logout`. The man pages for these programs document which files they use.

For `bash`, it is useful to activate "automatic completion" in the `/etc/bash.bashrc` file (simply uncomment a few lines).

BACK TO BASICS

**Automatic completion**

Many command interpreters provide a completion feature, which allows the shell to automatically complete a partially typed command name or argument when the user hits the Tab key. This lets users work more efficiently and be less error-prone.

This function is very powerful and flexible. It is possible to configure its behavior according to each command. Thus, the first argument following `apt` will be proposed according to the syntax of this command, even if it does not match any file (in this case, the possible choices are `install`, `remove`, `upgrade`, etc.).

The package *bash-completion* contains completions for most common programs.

BACK TO BASICS

**The tilde, a shortcut to HOME**

The tilde is often used to indicate the directory to which the environment variable, `HOME`, points (being the user's home directory, such as `/home/rhertzog/`). Command interpreters automatically make the substitution: `~/hello.txt` becomes `/home/rhertzog/hello.txt`.

The tilde also allows access to another user's home directory. Thus, `~rmas/bonjour.txt` is synonymous with `/home/rmas/bonjour.txt`.

In addition to these common scripts, each user can create their own `~/.bashrc` and `~/.bash_profile` to configure their shell. The most common changes are the addition of aliases; these are words that are automatically replaced with the execution of a command, which makes it faster to invoke that command. For instance, you could create the `la` alias for the command `ls -la | less` command; then you only have to type `la` to inspect the contents of a directory in detail.

BACK TO BASICS

**Environment variables**

Environment variables allow storage of global settings for the shell or various other programs called. They are contextual (each process has its own set of environment variables) but inheritable. This last characteristic offers the possibility for a login shell to declare variables which will be passed down to all programs it executes.

Setting default environment variables is an important element of shell configuration. Leaving aside the variables specific to a shell, it is preferable to place them in the `/etc/environment` file, since it is used by the various programs likely to initiate a shell session. Variables typically defined there include `ORGANIZATION`, which usually contains the name of the company or organization, and `HTTP_PROXY`, which indicates the existence and location of an HTTP proxy.

TIP

**All shells configured identically**

Users often want to configure their login and interactive shells in the same way. To do this, they choose to interpret (or "source") the content from `~/.bashrc` in the `~/.bash_profile` file. It is possible to do the same with files common to all users (by calling `/etc/bash.bashrc` from `/etc/profile`).

## 8.7. Printer Configuration

Printer configuration used to cause a great many headaches for administrators and users alike. These headaches are now mostly a thing of the past, thanks to CUPS, the free print server using the IPP (Internet Printing Protocol).

Debian distributes CUPS divided between several packages. The heart of the system is the scheduler, cupsd, which is in the *cups-daemon* package. *cups-client* contains utility programs to interact with the server, cupsd. lpadmin is probably the most important utility, as it is crucial for setting up a printer, but there are also facilities to disable or enable a printer queue, view or delete print jobs and display or set printer options. The CUPS framework is based on the System V printing system, but there is a compatibility package, *cups-bsd*, allowing use of commands such as `lpr`, `lpq` and `lprm` from the traditional BSD printing system.

COMMUNITY
**CUPS**

CUPS is a project and a trademark owned and managed by Apple, Inc. Prior to its acquisition by Apple it was known as the Commmon Unix Printing System.

➨ https://www.cups.org/

The scheduler manages print jobs and these jobs traverse a filtering system to produce a file that the printer will understand and print. The filtering system is provided by the *cups-filters* (https://salsa.debian.org/printing-team/cups-filters) package in conjunction with *printer-driver-** packages. CUPS in combination with *cups-filters* and *printer-driver-** is the basis for the Debian printing system.

Modern printers manufactured and sold within the last ten years are nearly always AirPrint-capable, and CUPS and cups-filters on Debian *Buster* have everything which is needed to take advantage of this facility on the network. In essence, these printers are IPP printers and an excellent fit for a driverless printing system, reducing the system to CUPS plus cups-filters. A printer-driver package can be dispensed with, and non-free printing software from vendors like Canon and Brother is no longer required. A USB-connected printer can take advantage of a modern printer with the *ippusbxd* package.

The command `apt install cups` will install CUPS and cups-filters. It will also install the recommended *printer-driver-gutenprint* to provide a driver for a wide range of printers, but, unless the printer is being operated driverlessly, an alternative printer-driver might be needed for the particular device.

As a package recommended by *cups-daemon*, *cups-browsed* will be on the system and networked print queues, and modern printers can be automatically discovered and set up from their DNS-SD broadcasts (Bonjour). USB printers will have to be set up manually as described in the next paragraph.

The printing system is administered easily through a web interface accessible at the local address http://localhost:631/. There you can add and remove USB and network printers and administer most aspects of their behavior. Similar administration tasks can also be carried out

via the graphical interface provided by a desktop environment or the `system-config-printer` graphical interface (from the homonym Debian package).

## 8.8. Configuring the Bootloader

It is probably already functional, but it is always good to know how to configure and install the bootloader in case it disappears from the Master Boot Record. This can occur after installation of another operating system, such as Windows. The following information can also help you to modify the bootloader configuration if needed.

BACK TO BASICS

**Master boot record**

The Master Boot Record (MBR) occupies the first 512 bytes of the first hard disk, and is the first thing loaded by the BIOS to hand over control to a program capable of booting the desired operating system. In general, a bootloader gets installed in the MBR, removing its previous content.

### 8.8.1. Identifying the Disks

CULTURE

***udev* and `/dev/`**

The `/dev/` directory traditionally houses so-called "special" files, intended to represent system peripherals (see sidebar "Device access permissions" page 176). Once upon a time, it used to contain all special files that could potentially be used. This approach had a number of drawbacks among which the fact that it restricted the number of devices that one could use (due to the hardcoded list of names), and that it was impossible to know which special files were actually useful.

Nowadays, the management of special files is entirely dynamic and matches better the nature of hot-swappable computer devices. The kernel cooperates with *udev* (section 9.11.3, "How *udev* Works" page 231) to create and delete them as needed when the corresponding devices appear and disappear. For this reason, `/dev/` doesn't need to be persistent and is thus a RAM-based filesystem that starts empty and contains only the relevant entries.

The kernel communicates lots of information about any newly added device and hands out a pair of major/minor numbers to identify it. With this `udevd` can create the special file under the name and with the permissions that it wants. It can also create aliases and perform additional actions (such as initialization or registration tasks). `udevd`'s behavior is driven by a large set of (customizable) rules.

With dynamically assigned names, you can thus keep the same name for a given device, regardless of the connector used or the connection order, which is especially useful when you use various USB peripherals. The first partition on the first hard drive can then be called `/dev/sda1` for backwards compatibility, or `/dev/root-partition` if you prefer, or even both at the same time since `udevd` can be configured to automatically create a symbolic link.

In ancient times, some kernel modules did automatically load when you tried to access the corresponding device file. This is no longer the case, and the peripheral's special file no longer exists prior to loading the module; this is no big deal, since most modules are loaded on boot thanks to automatic hardware detection. But for undetectable peripherals (such as very old disk drives or PS/2 mice), this doesn't

work. Consider adding the modules, `floppy`, `psmouse` and `mousedev` to `/etc/modules` in order to force loading them on boot.

Configuration of the bootloader must identify the different hard drives and their partitions. Linux uses "block" special files stored in the `/dev/` directory, for this purpose. Since Debian *Squeeze*, the naming scheme for hard drives has been unified by the Linux kernel, and all hard drives (IDE/PATA, SATA, SCSI, USB, IEEE 1394) are now represented by `/dev/sd*`.

Each partition is represented by its number on the disk on which it resides: for instance, `/dev/sda1` is the first partition on the first disk, and `/dev/sdb3` is the third partition on the second disk.

The PC architecture (or "i386", including its younger cousin "amd64") has long been limited to using the "MS-DOS" partition table format, which only allows four "primary" partitions per disk. To go beyond this limitation under this scheme, one of them has to be created as an "extended" partition, and it can then contain additional "secondary" partitions. These secondary partitions are numbered from 5. Thus the first secondary partition could be `/dev/sda5`, followed by `/dev/sda6`, etc.

Another restriction of the MS-DOS partition table format is that it only allows disks up to 2 TiB in size, which is becoming a real problem with recent disks.

A new partition table format called GPT loosens these constraints on the number of partitions (it allows up to 128 partitions when using standard settings) and on the size of the disks (up to 8 ZiB, which is more than 8 billion terabytes). If you intend to create many physical partitions on the same disk, you should therefore ensure that you are creating the partition table in the GPT format when partitioning your disk.

It is not always easy to remember what disk is connected to which SATA controller, or in third position in the SCSI chain, especially since the naming of hotplugged hard drives (which includes among others most SATA disks and external disks) can change from one boot to another. Fortunately, `udev` creates, in addition to `/dev/sd*`, symbolic links with a fixed name, which you could then use if you wished to identify a hard drive in a non-ambiguous manner. These symbolic links are stored in `/dev/disk/by-id`. On a machine with two physical disks, for example, one could find the following:

```
mirexpress:/dev/disk/by-id# ls -l
total 0
lrwxrwxrwx 1 root root  9 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP -> ../../sda
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP-part2 -> ../../sda2
[...]
lrwxrwxrwx 1 root root  9 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697 ->
       ../../sdb
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697-
       part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2_WD-WCAT00241697-
       part2 -> ../../sdb2
```

```
[...]
lrwxrwxrwx 1 root root  9 23 jul. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP -> ../../sda
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP-part1 ->
       ../../sda1
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_STM3500418AS_9VM3L3KP-part2 ->
       ../../sda2
[...]
lrwxrwxrwx 1 root root  9 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697 ->
       ../../sdb
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697-
       part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-_WD-WCAT00241697-
       part2 -> ../../sdb2
[...]
lrwxrwxrwx 1 root root  9 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0 ->
       ../../sdc
lrwxrwxrwx 1 root root 10 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0-part1 ->
        ../../sdc1
lrwxrwxrwx 1 root root 10 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e26ccc11a-0:0-part2 ->
        ../../sdc2
[...]
lrwxrwxrwx 1 root root  9 23 jul. 08:58 wwn-0x5000c50015c4842f -> ../../sda
lrwxrwxrwx 1 root root 10 23 jul. 08:58 wwn-0x5000c50015c4842f-part1 -> ../../sda1
[...]
mirexpress:/dev/disk/by-id#
```

Note that some disks are listed several times (because they behave simultaneously as ATA disks and SCSI disks), but the relevant information is mainly in the model and serial numbers of the disks, from which you can find the peripheral file.

The example configuration files given in the following sections are based on the same setup: a single SATA disk, where the first partition is an old Windows installation and the second contains Debian GNU/Linux.

### 8.8.2. Configuring LILO

*LILO* (LInux LOader) is the oldest bootloader — solid but rustic. It writes the physical address of the kernel to boot on the MBR, which is why each update to LILO (or its configuration file) must be followed by the command `lilo`. Forgetting to do so will render a system unable to boot if the old kernel was removed or replaced as the new one will not be in the same location on the disk.

LILO's configuration file is `/etc/lilo.conf`; a simple file for standard configuration is illustrated in the example below.

**Example 8.4** *LILO configuration file*

```
# The disk on which LILO should be installed.
# By indicating the disk and not a partition.
# you order LILO to be installed on the MBR.
boot=/dev/sda
# the partition that contains Debian
root=/dev/sda2
# the item to be loaded by default
default=Linux

# the most recent kernel image
image=/vmlinuz
  label=Linux
  initrd=/initrd.img
  read-only

# Old kernel (if the newly installed kernel doesn't boot)
image=/vmlinuz.old
  label=LinuxOLD
  initrd=/initrd.img.old
  read-only
  optional

# only for Linux/Windows dual boot
other=/dev/sda1
  label=Windows
```

### 8.8.3. GRUB 2 Configuration

*GRUB* (GRand Unified Bootloader) is more recent. It is not necessary to invoke it after each update of the kernel; *GRUB* knows how to read the filesystems and find the position of the kernel on the disk by itself. To install it on the MBR of the first disk, simply type `grub-install /dev/sda`.

NOTE

**Disk names for GRUB**

GRUB can only identify hard drives based on information provided by the BIOS. `(hd0)` corresponds to the first disk thus detected, `(hd1)` the second, etc. In most cases, this order corresponds exactly to the usual order of disks under Linux, but problems can occur when you associate SCSI and IDE disks. GRUB used to store the correspondences that it detects in the file `/boot/grub/device.map`, GRUB avoids this problem nowadays by using UUIDs or file system labels when generating `grub.cfg`. However, the device map file is not obsolete yet, since it can be used to override when the current environment is different from the one on boot. If you find errors there (because you know that your BIOS detects drives in a different order), correct them manually and run `grub-install` again. `grub-mkdevicemap` can help creating a `device.map` file from which to start.

Partitions also have a specific name in GRUB. When you use "classical" partitions in MS-DOS format, the first partition on the first disk is labeled, `(hd0,msdos1)`, the second `(hd0,msdos2)`, etc.

GRUB 2 configuration is stored in `/boot/grub/grub.cfg`, but this file (in Debian) is generated from others. Be careful not to modify it by hand, since such local modifications will be lost the next time `update-grub` is run (which may occur upon update of various packages). The most common modifications of the `/boot/grub/grub.cfg` file (to add command line parameters to the kernel or change the duration that the menu is displayed, for example) are made through the variables in `/etc/default/grub`. To add entries to the menu, you can either create a `/boot/grub/custom.cfg` file or modify the `/etc/grub.d/40_custom` file. For more complex configurations, you can modify other files in `/etc/grub.d`, or add to them; these scripts should return configuration snippets, possibly by making use of external programs. These scripts are the ones that will update the list of kernels to boot: `10_linux` takes into consideration the installed Linux kernels; `20_linux_xen` takes into account Xen virtual systems, and `30_os-prober` lists other operating systems (Windows, OS X, Hurd).

## 8.9. Other Configurations: Time Synchronization, Logs, Sharing Access...

The many elements listed in this section are good to know for anyone who wants to master all aspects of configuration of the GNU/Linux system. They are, however, treated briefly and frequently refer to the documentation.

### 8.9.1. Timezone

BACK TO BASICS

**Symbolic links**

A symbolic link is a pointer to another file. When you access it, the file to which it points is opened. Removal of the link will not cause deletion of the file to which it points. Likewise, it does not have its own set of permissions, but rather retains the permissions of its target. Finally, it can point to any type of file: directories, special files (sockets, named pipes, device files, etc.), even other symbolic links.

The `ln -s` *`target link-name`* command creates a symbolic link, named *link-name*, pointing to *target*.

If the target does not exist, then the link is "broken" and accessing it will result in an error indicating that the target file does not exist. If the link points to another link, you will have a "chain" of links that turns into a "cycle" if one of the targets points to one of its predecessors. In this case, accessing one of the links in the cycle will result in a specific error ("too many levels of symbolic links"); this means the kernel gave up after several rounds of the cycle.

The timezone, configured during initial installation, is a configuration item for the *tzdata* package. To modify it, use the `dpkg-reconfigure tzdata` command, which allows you to choose the timezone to be used in an interactive manner. Its configuration is stored in the `/etc/timezone` file. Additionally, the corresponding file in the `/usr/share/zoneinfo` directory is copied into `/etc/localtime`; this file contains the rules governing the dates where daylight saving time is active, for countries that use it.

When you need to temporarily change the timezone, use the `TZ` environment variable, which takes priority over the configured system default:

```
$ date
Thu Feb 19 11:25:18 CET 2015
$ TZ="Pacific/Honolulu" date
Thu Feb 19 00:25:21 HST 2015
```

NOTE

**System clock, hardware clock**

There are two time sources in a computer. A computer's motherboard has a hardware clock, called the "CMOS clock". This clock is not very precise, and provides rather slow access times. The operating system kernel has its own, the software clock, which it keeps up to date with its own means (possibly with the help of time servers, see section 8.9.2, "Time Synchronization" page 184). This system clock is generally more accurate, especially since it doesn't need access to hardware variables. However, since it only exists in live memory, it is zeroed out every time the machine is booted, contrary to the CMOS clock, which has a battery and therefore "survives" rebooting or halting of the machine. The system clock is, thus, set from the CMOS clock during boot, and the CMOS clock is updated on shutdown (to take into account possible changes or corrections if it has been improperly adjusted).

In practice, there is a problem, since the CMOS clock is nothing more than a counter and contains no information regarding the time zone. There is a choice to make regarding its interpretation: either the system considers it runs in universal time (UTC, formerly GMT), or in local time. This choice could be a simple shift, but things are actually more complicated: as a result of daylight saving time, this offset is not constant. The result is that the system has no way to determine whether the offset is correct, especially around periods of time change. Since it is always possible to reconstruct local time from universal time and the timezone information, we strongly recommend using the CMOS clock in universal time.

Unfortunately, Windows systems in their default configuration ignore this recommendation; they keep the CMOS clock on local time, applying time changes when booting the computer by trying to guess during time changes if the change has already been applied or not. This works relatively well, as long as the system has only Windows running on it. But when a computer has several systems (whether it be a "dual-boot" configuration or running other systems via virtual machine), chaos ensues, with no means to determine if the time is correct. If you absolutely must retain Windows on a computer, you should either configure it to keep the CMOS clock as UTC (setting the registry key `HKLM\SYSTEM\CurrentControlSet\Control\TimeZoneInformation\RealTimeIsUniversal` to "1" as a DWORD), or use `hwclock --localtime --set` on the Debian system to set the hardware clock and mark it as tracking the local time (and make sure to manually check your clock in spring and autumn).

### 8.9.2. Time Synchronization

Time synchronization, which may seem superfluous on a computer, is very important on a network. Since users do not have permissions allowing them to modify the date and time, it is important for this information to be precise to prevent confusion. Furthermore, having all of the computers on a network synchronized allows better cross-referencing of information from logs on different machines. Thus, in the event of an attack, it is easier to reconstruct the chronological sequence of actions on the various machines involved in the compromise. Data collected

on several machines for statistical purposes won't make a great deal of sense if they are not synchronized.

BACK TO BASICS

**NTP**

NTP (Network Time Protocol) allows a machine to synchronize with others fairly accurately, taking into consideration the delays induced by the transfer of information over the network and other possible offsets.

While there are numerous NTP servers on the Internet, the more popular ones may be overloaded. This is why we recommend using the *pool.ntp.org* NTP server, which is, in reality, a group of machines that have agreed to serve as public NTP servers. You could even limit use to a sub-group specific to a country, with, for example, *us.pool.ntp.org* for the United States, or *ca.pool.ntp.org* for Canada, etc.

However, if you manage a large network, it is recommended that you install your own NTP server, which will synchronize with the public servers. In this case, all the other machines on your network can use your internal NTP server instead of increasing the load on the public servers. You will also increase homogeneity with your clocks, since all the machines will be synchronized on the same source, and this source is very close in terms of network transfer times.

### *For Workstations*

Since work stations are regularly rebooted (even if only to save energy), synchronizing them by NTP at boot is enough. To do so, simply install the *ntpdate* package. You can change the NTP server used if needed by modifying the `/etc/default/ntpdate` file.

### *For Servers*

Servers are only rarely rebooted, and it is very important for their system time to be correct. To permanently maintain correct time, you would install a local NTP server, a service offered in the *ntp* package. In its default configuration, the server will synchronize with *pool.ntp.org* and provide time in response to requests coming from the local network. You can configure it by editing the `/etc/ntp.conf` file, the most significant alteration being the NTP server to which it refers. If the network has a lot of servers, it may be interesting to have one local time server which synchronizes with the public servers and is used as a time source by the other servers of the network.

GOING FURTHER

**GPS modules and other time sources**

If time synchronization is particularly crucial to your network, it is possible to equip a server with a GPS module (which will use the time from GPS satellites) or a DCF-77 module (which will sync time with the atomic clock near Frankfurt, Germany). In this case, the configuration of the NTP server is a little more complicated, and prior consultation of the documentation is an absolute necessity.

### 8.9.3. Rotating Log Files

Log files can grow, fast, and it is necessary to archive them. The most common scheme is a rotating archive: the log file is regularly archived, and only the latest *X* archives are retained. `logrotate`, the program responsible for these rotations, follows directives given in the `/etc/logrotate.conf` file and all of the files in the `/etc/logrotate.d/` directory. The administrator may modify these files, if they wish to adapt the log rotation policy defined by Debian. The `logrotate(1)` man page describes all of the options available in these configuration files. You may want to increase the number of files retained in log rotation, or move the log files to a specific directory dedicated to archiving them rather than delete them. You could also send them by e-mail to archive them elsewhere.

The `logrotate` program is executed daily by the `cron` scheduling program (described in section 9.7, "Scheduling Tasks with `cron` and `atd`" page 222).

### 8.9.4. Sharing Administrator Rights

Frequently, several administrators work on the same network. Sharing the root passwords is not very elegant, and opens the door for abuse due to the anonymity such sharing creates. The solution to this problem is the `sudo` program, which allows certain users to execute certain commands with special rights. In the most common use case, `sudo` allows a trusted user to execute any command as root. To do so, the user simply executes `sudo` *`command`* and authenticates using their personal password.

When installed, the *sudo* package gives full root rights to members of the `sudo` Unix group. To delegate other rights, the administrator must use the `visudo` command, which allows them to modify the `/etc/sudoers` configuration file (here again, this will invoke the `vi` editor, or any other editor indicated in the `EDITOR` environment variable). Adding a line with *username* ALL=(ALL) ALL allows the user in question to execute any command as root.

More sophisticated configurations allow authorization of only specific commands to specific users. All the details of the various possibilities are given in the `sudoers(5)` man page.

### 8.9.5. List of Mount Points

BACK TO BASICS

**Mounting and unmounting**

In a Unix-like system such as Debian, files are organized in a single tree-like hierarchy of directories. The `/` directory is called the "root directory"; all additional directories are sub-directories within this root. "Mounting" is the action of including the content of a peripheral device (often a hard drive) into the system's general file tree. As a consequence, if you use a separate hard drive to store users' personal data, this disk will have to be "mounted" in the `/home/` directory. The root filesystem is always mounted at boot by the kernel; other devices are often mounted later during the startup sequence or manually with the `mount` command.

Some removable devices are automatically mounted when connected, especially when using the GNOME, Plasma or other graphical desktop environments. Others have to be mounted manually by the user. Likewise, they must be unmounted

(removed from the file tree). Normal users do not usually have permission to execute the `mount` and `umount` commands. The administrator can, however, authorize these operations (independently for each mount point) by including the `user` option in the `/etc/fstab` file.

The `mount` command can be used without arguments to list all mounted filesystems; you can execute `findmnt --fstab` to show only the filesystems from `/etc/fstab`. The following parameters are required to mount or unmount a device. For the complete list, please refer to the corresponding man pages, `mount(8)` and `umount(8)`. For simple cases, the syntax is simple too: for example, to mount the `/dev/sdc1` partition, which has an ext3 filesystem, into the `/mnt/tmp/` directory, you would simply run `mount -t ext3 /dev/sdc1 /mnt/tmp/`.

The `/etc/fstab` file gives a list of all possible mounts that happen either automatically on boot or manually for removable storage devices. Each mount point is described by a line with several space-separated fields:

- file system: this indicates where the filesystem to be mounted can be found, it can be a local device (hard drive partition, CD-ROM) or a remote filesystem (such as NFS).

  This field is frequently replaced with the unique ID of the filesystem (which you can determine with `blkid` **`device`**) prefixed with UUID=. This guards against a change in the name of the device in the event of addition or removal of disks, or if disks are detected in a different order.
- mount point: this is the location on the local filesystem where the device, remote system, or partition will be mounted.
- type: this field defines the filesystem used on the mounted device. ext4, ext3, vfat, ntfs, btrfs, xfs are a few examples.

  BACK TO BASICS
  **NFS, a network filesystem**
  NFS is a network filesystem; under Linux, it allows transparent access to remote files by including them in the local filesystem.

  A complete list of known filesystems is available in the `mount(8)` man page. The swap special value is for swap partitions; the `auto` special value tells the `mount` program to automatically detect the filesystem (which is especially useful for disk readers and USB keys, since each one might have a different filesystem);
- options: there are many of them, depending on the filesystem, and they are documented in the `mount` man page. The most common are
  - `rw` or `ro`, meaning, respectively, that the device will be mounted with read/write or read-only permissions.
  - `noauto` deactivates automatic mounting on boot.
  - `nofail` allows the boot to proceed even when the device is not present. Make sure to put this option for external drives that might be unplugged when you boot, because `systemd` really ensures that all mount points that must be automatically mounted are actually mounted before letting the boot process continue to its end. Note that you can combine this with `x-systemd.device-timeout=5s` to tell `systemd` to not wait more than 5 seconds for the device to appear (see `systemd.mount(5)`).

  - user authorizes all users to mount this filesystem (an operation which would otherwise be restricted to the root user).
  - defaults means the group of default options: rw, suid, dev, exec, auto, nouser and async, each of which can be individually disabled after defaults by adding nosuid, nodev and so on to block suid, dev and so on. Adding the user option reactivates it, since defaults includes nouser.

- dump: this field is almost always set to 0. When it is 1, it tells the `dump` tool that the partition contains data that is to be backed up.
- pass: this last field indicates whether the integrity of the filesystem should be checked on boot, and in which order this check should be executed. If it is 0, no check is conducted. The root filesystem should have the value 1, while other permanent filesystems get the value 2.

**Example 8.5** *Example `/etc/fstab` file*

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/sda1 during installation
UUID=c964222e-6af1-4985-be04-19d7c764d0a7 / ext3 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=ee880013-0f63-4251-b5c6-b771f53bd90e none swap sw  0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto 0       0
/dev/fd0        /media/floppy   auto    rw,user,noauto  0       0
arrakis:/shared /shared         nfs     defaults        0       0
```

The last entry in this example corresponds to a network filesystem (NFS): the `/shared/` directory on the *arrakis* server is mounted at `/shared/` on the local machine. The format of the `/etc/fstab` file is documented on the `fstab(5)` man page.

GOING FURTHER

**Auto-mounting**

systemd is able to manage automount points: those are filesystems that are mounted on-demand when a user attempts to access their target mount points. It can also unmount these filesystems when no process is accessing them any longer.

Like most concepts in systemd, automount points are managed with dedicated units (using the `.automount` suffix). See `systemd.automount(5)` for their precise syntax.

Other auto-mounting utilities exist, such as `automount` in the *autofs* package or `amd` in the *am-utils*.

Note also that GNOME, Plasma, and other graphical desktop environments work together with *udisks*, and can automatically mount removable media when they are connected.

### 8.9.6. `locate` and `updatedb`

The `locate` command can find the location of a file when you only know part of the name. It sends a result almost instantaneously, since it consults a database that stores the location of all the files on the system; this database is updated daily by the `updatedb` command. There are multiple implementations of the `locate` command and Debian picked *mlocate* for its standard system.

`mlocate` is smart enough to only return files which are accessible to the user running the command even though it uses a database that knows about all files on the system (since its `updatedb` implementation runs with root rights). For extra safety, the administrator can use `PRUNEDPATHS` in `/etc/updatedb.conf` to exclude some directories from being indexed.

## 8.10. Compiling a Kernel

The kernels provided by Debian include the largest possible number of features, as well as the maximum of drivers, in order to cover the broadest spectrum of existing hardware configurations. This is why some users prefer to recompile the kernel in order to only include what they specifically need. There are two reasons for this choice. First, it may be to optimize memory consumption, since the kernel code, even if it is never used, occupies memory for nothing (and never "goes down" on the swap space, since it is actual RAM that it uses), which can decrease overall system performance. A locally compiled kernel can also limit the risk of security problems since only a fraction of the kernel code is compiled and run.

NOTE

**Security updates**

If you choose to compile your own kernel, you must accept the consequences: Debian cannot ensure security updates for your custom kernel. By keeping the kernel provided by Debian, you benefit from updates prepared by the Debian Project's security team.

Recompilation of the kernel is also necessary if you want to use certain features that are only available as patches (and not included in the standard kernel version).

GOING FURTHER

**The Debian Kernel Handbook**

The Debian kernel team maintains the "Debian Kernel Handbook" (also available in the *debian-kernel-handbook* package) with comprehensive documentation about most kernel related tasks and about how official Debian kernel packages are handled. This is the first place you should look into if you need more information than what is provided in this section.

➨ `https://kernel-team.pages.debian.net/kernel-handbook/`

### 8.10.1. Introduction and Prerequisites

Unsurprisingly Debian manages the kernel in the form of a package, which is not how kernels have traditionally been compiled and installed. Since the kernel remains under the control of

the packaging system, it can then be removed cleanly, or deployed on several machines. Furthermore, the scripts associated with these packages automate the interaction with the bootloader and the initrd generator.

The upstream Linux sources contain everything needed to build a Debian package of the kernel. But you still need to install *build-essential* to ensure that you have the tools required to build a Debian package. Furthermore, the configuration step for the kernel requires the *libncurses5-dev* package. Finally, the *fakeroot* package will enable creation of the Debian package without using administrator's rights.

CULTURE

**The good old days of *kernel-package***

Before the Linux build system gained the ability to build proper Debian packages, the recommended way to build such packages was to use `make-kpkg` from the *kernel-package* package.

### 8.10.2. Getting the Sources

Like anything that can be useful on a Debian system, the Linux kernel sources are available in a package. To retrieve them, just install the *linux-source*-version package. The `apt search ^linux-source` command lists the various kernel versions packaged by Debian. The latest version is available in the *Unstable* distribution: you can retrieve them without much risk (especially if your APT is configured according to the instructions of section 6.2.6, "Working with Several Distributions" page 124). Note that the source code contained in these packages does not correspond precisely with that published by Linus Torvalds and the kernel developers; like all distributions, Debian applies a number of patches, which might (or might not) find their way into the upstream version of Linux. These modifications include backports of fixes/features/drivers from newer kernel versions, new features not yet (entirely) merged in the upstream Linux tree, and sometimes even Debian specific changes.

The remainder of this section focuses on the 4.19 version of the Linux kernel, but the examples can, of course, be adapted to the particular version of the kernel that you want.

We assume the *linux-source-4.19* package has been installed. It contains `/usr/src/linux-source-4.19.tar.xz`, a compressed archive of the kernel sources. You must extract these files in a new directory (not directly under `/usr/src/`, since there is no need for special permissions to compile a Linux kernel): `~/kernel/` is appropriate.

```
$ mkdir ~/kernel; cd ~/kernel
$ tar -xaf /usr/src/linux-source-4.19.tar.xz
```

CULTURE

**Location of kernel sources**

Traditionally, Linux kernel sources would be placed in `/usr/src/linux/` thus requiring root permissions for compilation. However, working with administrator rights should be avoided when not needed. There is a `src` group that allows members to work in this directory, but working in `/usr/src/` should be avoided, nevertheless. By keeping the kernel sources in a personal directory, you get security on all counts: no files in `/usr/` unknown to the packaging system, and no risk of misleading programs that read `/usr/src/linux` when trying to gather information on the used kernel.

### 8.10.3. Configuring the Kernel

The next step consists of configuring the kernel according to your needs. The exact procedure depends on the goals.

When recompiling a more recent version of the kernel (possibly with an additional patch), the configuration will most likely be kept as close as possible to that proposed by Debian. In this case, and rather than reconfiguring everything from scratch, it is sufficient to copy the `/boot/config-`*`version`* file (the version is that of the kernel currently used, which can be found with the `uname -r` command) into a `.config` file in the directory containing the kernel sources.

```
$ cp /boot/config-4.19.0-5-amd64 ~/kernel/linux-source-4.19/.config
```

Unless you need to change the configuration, you can stop here and skip to section 8.10.4, "Compiling and Building the Package" page 192. If you need to change it, on the other hand, or if you decide to reconfigure everything from scratch, you must take the time to configure your kernel. There are various dedicated interfaces in the kernel source directory that can be used by calling the `make` *`target`* command, where *target* is one of the values described below.

`make menuconfig` compiles and executes a text-mode interface (this is where the *libncurses5-dev* package is required) which allows navigating the options available in a hierarchical structure. Pressing the Space key changes the value of the selected option, and Enter validates the button selected at the bottom of the screen; Select returns to the selected sub-menu; Exit closes the current screen and moves back up in the hierarchy; Help will display more detailed information on the role of the selected option. The arrow keys allow moving within the list of options and buttons. To exit the configuration program, choose Exit from the main menu. The program then offers to save the changes you've made; accept if you are satisfied with your choices.

Other interfaces have similar features, but they work within more modern graphical interfaces; such as `make xconfig` which uses a Qt graphical interface, and `make gconfig` which uses GTK+. The former requires *libqt4-dev*, while the latter depends on *libglade2-dev* and *libgtk2.0-dev*.

When using one of those configuration interfaces, it is always a good idea to start from a reasonable default configuration. The kernel provides such configurations in `arch/`*`arch`*`/configs/*_defconfig` and you can put your selected configuration in place with a command like `make x86_64_defconfig` (in the case of a 64-bit PC) or `make i386_defconfig` (in the case of a 32-bit PC).

TIP

**Dealing with outdated `.config` files**

When you provide a `.config` file that has been generated with another (usually older) kernel version, you will have to update it. You can do so with `make oldconfig`, it will interactively ask you the questions corresponding to the new configuration options. If you want to use the default answer to all those questions you can use `make olddefconfig`. With `make oldnoconfig`, it will assume a negative answer to all questions.

### 8.10.4. Compiling and Building the Package

NOTE

**Clean up before rebuilding**

If you have already compiled once in the directory and wish to rebuild everything from scratch (for example, because you substantially changed the kernel configuration), you will have to run `make clean` to remove the compiled files. `make distclean` removes even more generated files, including your `.config` file too, so make sure to backup it first. If you copied the configuration from `/boot/`, you must change the system trusted keys option, providing an empty string is enough: `CONFIG_SYSTEM_TRUSTED_KEYS = ""`.

Once the kernel configuration is ready, a simple `make deb-pkg` will generate up to 5 Debian packages: *linux-image*-version that contains the kernel image and the associated modules, *linux-headers*-version which contains the header files required to build external modules, *linux-firmware-image*-version which contains the firmware files needed by some drivers (this package might be missing when you build from the kernel sources provided by Debian), *linux-image*-version-*dbg* which contains the debugging symbols for the kernel image and its modules, and *linux-libc-dev* which contains headers relevant to some user-space libraries like GNU glibc.

The *version* is defined by the concatenation of the upstream version (as defined by the variables VERSION, PATCHLEVEL, SUBLEVEL and EXTRAVERSION in the `Makefile`), of the LOCALVERSION configuration parameter, and of the LOCALVERSION environment variable. The package version reuses the same version string with an appended revision that is regularly incremented (and stored in `.version`), except if you override it with the KDEB_PKGVERSION environment variable.

```
$ make deb-pkg LOCALVERSION=-falcot KDEB_PKGVERSION=$(make kernelversion)-1
[...]
$ ls ../*.deb
../linux-headers-4.19.37-falcot_4.19.37-1_amd64.deb
../linux-image-4.19.37-falcot_4.19.37-1_amd64.deb
../linux-libc-dev_4.19.37-1_amd64.deb
```

### 8.10.5. Compiling External Modules

Some modules are maintained outside of the official Linux kernel. To use them, they must be compiled alongside the matching kernel. A number of common third party modules are provided by Debian in dedicated packages, such as *vpb-driver-source* (extra modules for Voicetronix telefony hardware) or *leds-alix-source* (driver of PCEngines ALIX 2/3 boards).

These packages are many and varied, `apt-cache rdepends module-assistant$` can show the list provided by Debian. However, a complete list isn't particularly useful since there is no particular reason for compiling external modules except when you know you need it. In such cases, the device's documentation will typically detail the specific module(s) it needs to function under Linux.

For example, let's look at the *dahdi-source* package: after installation, a `.tar.bz2` of the module's sources is stored in `/usr/src/`. While we could manually extract the tarball and build the module, in practice we prefer to automate all this using DKMS. Most modules offer the required DKMS integration in a package ending with a -dkms suffix. In our case, installing *dahdi-dkms* is all that is needed to compile the kernel module for the current kernel provided that we have the *linux-headers-** package matching the installed kernel. For instance, if you use *linux-image-amd64*, you would also install *linux-headers-amd64*.

```
$ sudo apt install dahdi-dkms

[...]
Setting up xtables-addons-dkms (2.12-0.1) ...
Loading new xtables-addons-2.12 DKMS files...
Building for 4.19.0-5-amd64
Building initial module for 4.19.0-5-amd64
Done.

dahdi_dummy.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/4.19.0-5-amd64/updates/dkms/
[...]
DKMS: install completed.
$ sudo dkms status
dahdi, DEB_VERSION, 4.19.0-5-amd64, x86_64: installed
$ sudo modinfo dahdi_dummy
filename:       /lib/modules/4.19.0-5-amd64/updates/dkms/dahdi_dummy.ko
license:        GPL v2
author:         Robert Pleh <robert.pleh@hermes.si>
description:    Timing-Only Driver
[...]
```

ALTERNATIVE

**module-assistant**

Before DKMS, *module-assistant* was the simplest solution to build and deploy kernel modules. It can still be used, in particular for packages lacking DKMS integration: with a simple command like `module-assistant auto-install dadhi` (or `m-a a-i dahdi` for short), the modules are compiled for the current kernel, put in a new Debian package, and that package gets installed on the fly.

### 8.10.6. Applying a Kernel Patch

Some features are not included in the standard kernel due to a lack of maturity or to some disagreement with the kernel maintainers. Such features may be distributed as patches that anyone is then free to apply to the kernel sources.

Debian sometimes provides some of these patches in *linux-patch-** packages but they often don't make it into stable releases (sometimes for the very same reasons that they are not merged into the official upstream kernel). These packages install files in the `/usr/src/kernel-patches/` directory.

To apply one or more of these installed patches, use the `patch` command in the sources directory then start compilation of the kernel as described above.

```
$ cd ~/kernel/linux-source-4.9
$ make clean
$ zcat /usr/src/kernel-patches/diffs/grsecurity2/grsecurity-3.1-4.9.11-201702181444.
      patch.gz | patch -p1
```

Note that a given patch may not necessarily work with every version of the kernel; it is possible for `patch` to fail when applying them to kernel sources. An error message will be displayed and give some details about the failure; in this case, refer to the documentation available in the Debian package of the patch (in the `/usr/share/doc/linux-patch-*/` directory). In most cases, the maintainer indicates for which kernel versions their patch is intended.

## 8.11. Installing a Kernel

### 8.11.1. Features of a Debian Kernel Package

A Debian kernel package installs the kernel image (`vmlinuz-`*`version`*), its configuration (`config-`*`version`*) and its symbols table (`System.map-`*`version`*) in `/boot/`. The modules are installed in the `/lib/modules/`*`version`*`/` directory.

CULTURE

**The symbols table**

The symbols table helps developers understand the meaning of a kernel error message; without it, kernel "oopses" (an "oops" is the kernel equivalent of a segmentation fault for user-space programs, in other words messages generated following an invalid pointer dereference) only contain numeric memory addresses, which is useless information without the table mapping these addresses to symbols and function names.

The package's configuration scripts automatically generate an initrd image, which is a mini-system designed to be loaded in memory (hence the name, which stands for "init ramdisk") by the bootloader, and used by the Linux kernel solely for loading the modules needed to access the devices containing the complete Debian system (for example, the driver for SATA disks). Finally, the post-installation scripts update the symbolic links `/vmlinuz`, `/vmlinuz.old`, `/initrd.img` and `/initrd.img.old` so that they point to the latest two kernels installed, respectively, as well as the corresponding initrd images.

Most of those tasks are offloaded to hook scripts in the `/etc/kernel/*.d/` directories. For instance, the integration with `grub` relies on `/etc/kernel/postinst.d/zz-update-grub` and `/etc/kernel/postrm.d/zz-update-grub` to call `update-grub` when kernels are installed or removed.

### 8.11.2. Installing with dpkg

Using `apt` is so convenient that it makes it easy to forget about the lower-level tools, but the easiest way of installing a compiled kernel is to use a command such as `dpkg -i` *`package`*`.deb`, where *package*.deb is the name of a *linux-image* package such as `linux-image-4.19.37-falcot_1_amd64.deb`.

The configuration steps described in this chapter are basic and can lead both to a server system or a workstation, and it can be massively duplicated in semi-automated ways. However, it is not enough by itself to provide a fully configured system. A few pieces are still in need of configuration, starting with low-level programs known as the "Unix services".

## Keywords

System boot
Initscripts
SSH
Telnet
Rights
Permissions
Supervision
Inetd
Cron
Backup
Hotplug
PCMCIA
APM
ACPI