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`