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`).