13 projets à réaliser dès 10 ans

Programmer avec Raspberry Pi

EN S'AMUSANT

Installer et tester le Raspberry Pi

Programmer un Raspberry Pi avec Python ou Scratch

Concevoir des jeux, des sites Web et des applications

RICHARD WENTK

pour les nuls

Programmer avec Raspberry Pi en s'amusant

pour les nuls

Richard Wentk

FIRST Interactive

Programmer avec Raspberry Pi en s’amusant pour les Nuls

Maquette : Enredos e Legendas Unip. Lda
Traduction : Olivier Engler

Edition française publiée en accord avec Wiley Publishing, Inc.

12 avenue d’Italie 75013 Paris
Tél. : 01 44 16 09 00
Fax: 01 44 16 09 01
e-mail : firstinfo@efirst.com
Internet : www.editionsfirst.fr
ISBN : 978-2-412-02889-6
ISBN numérique : 9782412032923
Dépôt légal : août 2017

Ce livre numérique a été converti initialement au format EPUB par Isako www.isako.com à partir de l'édition papier du même ouvrage.

Introduction

Que sais-tu à propos des ordinateurs ? De nos jours, tout le monde ou presque sait lancer un jeu vidéo, regarder des films, écouter de la musique et se promener sur le Web, sans trop savoir comment tout cela fonctionne. L'ordinateur marche, et c'est tout ce qui nous intéresse.

Mais tu as sans doute envie d'en savoir un peu plus. Qu'est-ce qui se passe quand on clique, quand on frappe une touche, quand on clique un lien dans une page Web, quand on lance une application ?

Et même plus : comment fait-on pour créer une page Web, une application ou un jeu vidéo ?

Tout le monde n'est pas pour autant curieux de ce qui va pourtant influer sur nos vies comme aucune autre invention auparavant, et c'est bien ainsi.

Mais toi, cela t'intéresse et cela pose une vraie question : où trouver des réponses ? Ce livre t'invite à faire tes premiers pas, mais il te faut surtout vite apprendre à chercher et à trouver tes propres réponses, sans te limiter aux réponses des autres.

Vouloir comprendre les ordinateurs donne l'occasion de prendre plaisir à résoudre des énigmes et des puzzles, à faire un peu de mathématiques, à rédiger du code source et à réaliser des projets remarquables et utiles. C'est aussi un puissant moyen pour devenir autonome dans ton apprentissage de la théorie comme de la pratique.

Et même si tu ne comptes pas devenir programmeur, rédiger du code source te permet de vérifier comment tu t'en sors pour apprendre quelque chose de totalement nouveau.

Trouver des solutions après avoir bien posé les problèmes compte pour plus de la moitié du chemin à parcourir. Et ce que tu as trouvé peut être proposé aux autres en partage. Tu combines ainsi tes efforts à ceux des autres et tout le monde en bénéficie.

Ne considère pas ce livre comme un recueil d’exercices de niveau scolaire. Il n’y a pas d’examen. Le livre contient surtout des accompagnements pour tes premiers pas, de quoi apprendre les bases dans de nombreux domaines. Ce n’est pas une série de recettes à appliquer sans réfléchir. Parfois, le sujet te demandera même de chercher un bout de la solution tout seul sur le Web.

Certains des projets pourront te sembler trop difficiles. Dans ce cas, passe aux autres ; tu y reviendras plus tard.

Ou pas. L’essentiel est que tu en retires du plaisir, en créant des projets qui te passionnent, mieux encore, qui te surprennent !

Mais n’abandonne pas devant le premier obstacle en prétendant que tu ferais mieux de t’occuper autrement. Sache que tous ceux qui ont appris à programmer ont connu ce genre de moment de relâchement, de temps à autre.

Et voici un vrai secret : une fois que tu as découvert tout ce que tu peux faire avec un Raspberry Pi, tu n’auras plus de réticence à y consacrer du temps.

À propos de ce livre

Ce livre te propose une introduction aux possibilités du nano-ordinateur Raspberry Pi.

Il y a des gens qui prétendent que le Raspberry Pi a été conçu pour des enfants et qu'il est simple d'emploi. Ce n'est vrai qu'en partie. Le Raspi est facile d'accès d'une certaine façon, mais sous d'autres aspects, il est plus complexe qu'une machine sous macOS ou sous Windows.

Le Raspi est vraiment adapté à l'apprentissage de la programmation et à la découverte de l'informatique. Tu peux en toute liberté créer des programmes mêlant logiciel et matériel. Voici un aperçu rapide du contenu :

- ce qu'est le Raspberry Pi et les membres de sa famille ;
- les composants dont tu auras besoin dans le dernier chapitre ;
- comment connecter tous ces câbles ;
- comment faire pour télécharger et installer les logiciels dans leur plus récente version ;
- l'alimentation du Raspi ;
- les paramètres de configuration ;
- comment arrêter proprement le Raspi ;
- avantages et inconvénients du système Linux ;
- comment utiliser l'interface graphique du Bureau sous Linux Raspbian ;

- trouver un fichier avec le Gestionnaire de fichiers ;
- les répertoires de Linux et les droits d'accès ;
- l'utilisateur normal et le super-utilisateur root ;
- saisir des commandes Linux au clavier ;
- les possibilités de Scratch, et la création d'un jeu ;
- composer de la musique électronique avec le langage de Sonic Pi ;
- rédiger du code et dessiner avec un programme en langage Python ;
- installer un serveur Web ;
- se servir de Python pour contrôler ton avance en reprenant le contrôle du personnage dans la version Pi de Minecraft ;
- connecter et exploiter une caméra ;
- réaliser un détecteur de mouvement.

Des prérequis minimalistes

Ce livre présuppose quelques connaissances et compétences de base :

- Tu sais utiliser un PC sous Windows ou un Mac.
- Tu sais te repérer dans le Bureau de ton interface graphique et le contrôler avec la souris et le clavier.
- Tu n’as pas peur de brancher des périphériques.
- Tu sais utiliser un moteur de recherche sur Internet.
- Tu as un petit peu d’argent à investir. Pour le circuit et ses câbles, compte moins de 50 euros. La caméra officielle te coûtera 30 euros de plus, et le capteur du dernier chapitre moins de 10 euros.

En revanche, tu n’as pas besoin d’avoir déjà abordé ni la programmation, ni l’électronique.

Les icônes de marge

Tout au long du livre, certains paragraphes comportent une icône dans la marge pour les distinguer. Voici ce que signifient ces icônes :

Ce genre de paragraphe offre une astuce ou un complément pour aider à la compréhension. Tout abus de lecture sera récompensé !

Cette icône marque un rappel d'un sujet déjà abordé qu'il vaut mieux ne pas oublier.

Ces textes te mettent en garde contre une possible erreur de compréhension ou de manipulation.

Cette icône distingue un complément d'information plus technique que tu peux ignorer dans un premier temps si le contenu te semble trop complexe.

Les fichiers des exemples du livre

Dans plusieurs chapitres, tu as à rédiger des instructions dans un langage informatique (Python, HTML ou CSS).

Bien que je te le déconseille, tu peux éviter de faire cette saisie en récupérant les fichiers des codes sources déjà écrits et testés. Il suffit d'aller sur le site de l'éditeur. :

`pourlesnuls.fr`

Dans la page d'accueil, clique l'icône de loupe pour chercher le titre de ce livre. Une fois sur la page du livre, clique le lien `Télécharger`. Tu obtiens un fichier d'archive que tu n'as plus qu'à déballer dans ton répertoire personnel `/home/pi`.

Par où commencer ?

Tu peux lire les chapitres dans n'importe quel ordre, mais tu profiteras d'une lecture de la première partie. Tu peux tout aussi bien picorer des techniques selon tes besoins. Tu peux aussi partir du sommaire puis passer tous les chapitres en série.

Tu peux aussi lire ce livre sans faire d'essai. Si tu es vraiment débutant, je te conseille de bien digérer la première partie.

Certains chapitres vers la fin supposent que tu as glané des informations dans les chapitres précédents. Ne tente pas les exercices avant d'avoir appris les bases !

Et surtout, bon voyage dans la joie de la découverte !

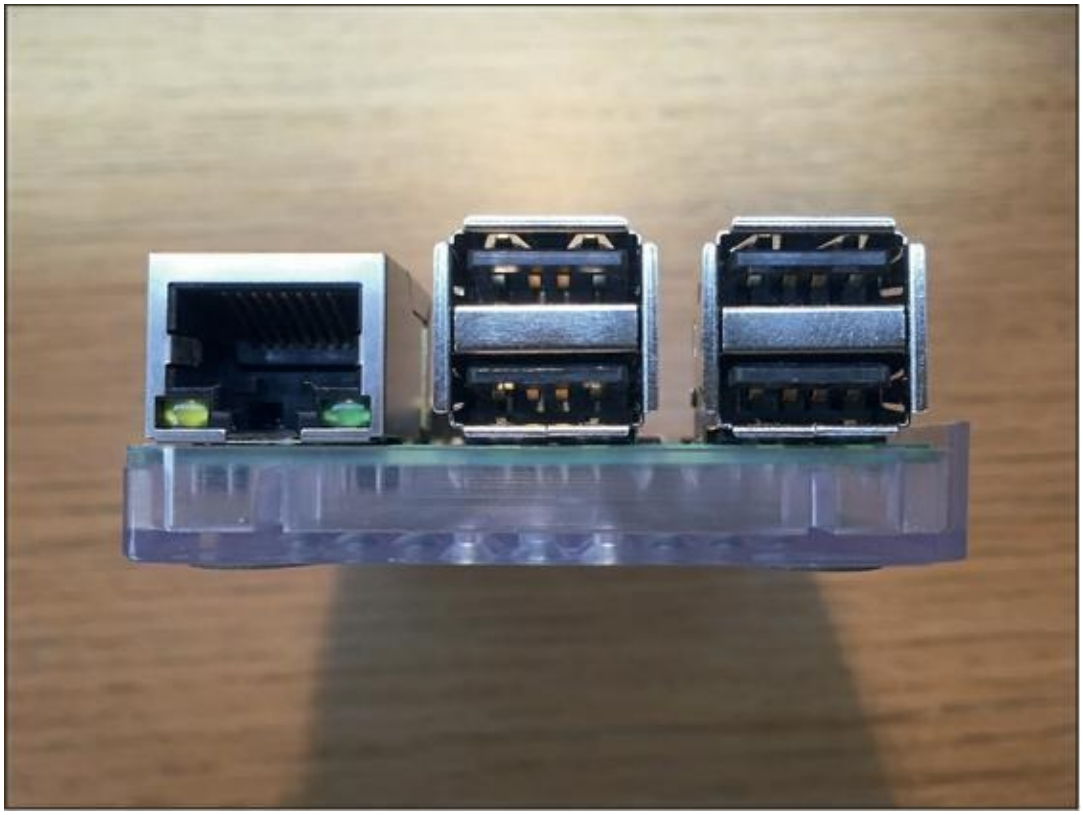
La mise en place

Au programme de cette partie :

Chapitre 1
Approvisionnement

Le Raspberry Pi, que je vais souvent abréger dans la suite Raspi, est un tout petit micro-ordinateur, c'est même un nano-ordinateur.

Pour être exact, il s'agit d'une toute petite carte mère, c'est-à-dire une carte électronique qui réunit un contrôleur et quelques circuits, notamment de la mémoire, et des connecteurs. Avant de pouvoir partir à l'aventure avec ton Raspi, il va falloir entourer ce circuit électronique de quelques périphériques, afin d'obtenir une véritable nano-machine complète.



Description du Raspberry Pi

La Figure 1.1 donne une vue générale du Raspberry Pi. Ce nano-ordinateur a été conçu au Royaume-Uni par la fondation Raspberry Pi (www.raspberrypi.org). La taille de la carte est proche de celle d'une carte bancaire, bien plus compacte que la carte-mère d'un PC ou d'un Mac. Et son prix est sans concurrence, puisqu'il est de l'ordre de 30 à 40 €.

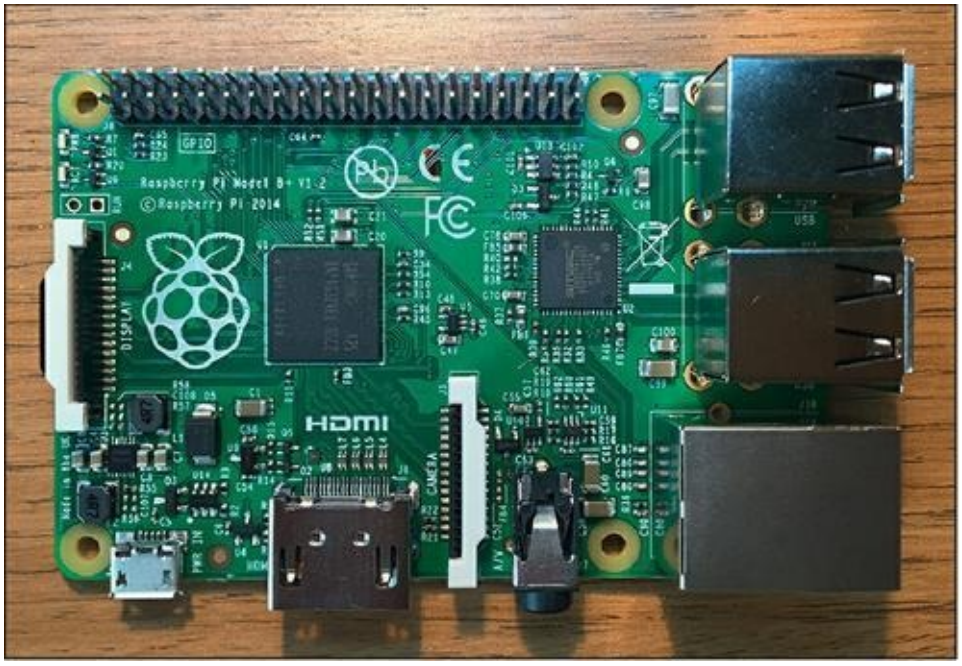


Figure 1.1 : Vue générale de la carte Raspberry Pi

Le suffixe Pi n'existe que pour le Raspberry. Tu ne trouveras pas d'Apple Pi ni de Myrtille Pi. En anglais, Raspberry signifie framboise, et Raspberry Pi fait penser à une tarte (*pie*) à la framboise.

Voici quelques-unes des choses que tu peux réaliser avec ce circuit :

- Apprendre comment fonctionne un ordinateur.
- Créer des jeux vidéo.
- Apprendre à écrire des programmes.

- Créer des pages Web.
- Composer de la musique électronique.
- Construire des projets électroniques simples.
- Créer des mondes Minecraft.
- Prendre beaucoup de plaisir à faire tout cela !

Ce qu'il est impossible de faire avec un Raspi

Le Raspi est un ordinateur complet, mais ce n'est pas un PC, ni même une tablette ou une console de jeu. Il n'offre évidemment pas autant de puissance. Voici quelques domaines dans lesquels le Raspi est limité :

- Tu ne peux pas utiliser Microsoft Windows ni aucun logiciel sous Windows.
- Tu ne peux pas installer des applications ou des jeux depuis un magasin tel que l'App store Apple.
- Tu ne peux pas créer des logiciels pour Windows, iOS ou macOS.
- Tu ne peux pas utiliser un navigateur tel que Chrome, Safari ou Internet Explorer.
- Tu ne peux pas utiliser les jeux vidéo les plus répandus.
- Mais que cette liste ne te perturbe pas. Voyons quelques points forts du circuit :

Possibilités d'un Raspi inaccessibles à un gros ordinateur

Tu peux faire bien des choses avec un Raspi qui sont impossibles avec un gros ordinateur. Par exemple :

- Tu peux en quelques minutes reconstruire tout le système de ton Raspi pour redémarrer à zéro en cas d'erreur.
- Tu peux apprendre à écrire tes propres logiciels sans risque.
- Tu peux créer des projets utiles et même les commercialiser.
- Tu peux étudier et modifier tous les logiciels fournis avec ton Raspi.
- Tu peux programmer ton Raspi pour qu'il déclenche des actions à certains moments où lorsqu'il détecte quelque chose avec un capteur.
- Tu peux facilement connecter au circuit différents capteurs : thermomètre, caméra, détecteur de proximité et autres.
- Tu peux laisser un programme fonctionner en permanence sans consommer beaucoup d'énergie.

L'histoire du Raspi

Le Raspberry Pi obéit à une tradition britannique. Dans les années 1980, le Royaume-Uni était pionnier mondial dans le domaine de la construction d'ordinateurs. C'était la grande époque du Sinclair ZX, du Dragon, du Tangerine et de l'Acorn. Ces machines étaient beaucoup moins puissantes qu'un Raspi, mais bien des adolescents ont appris à programmer avec ; certains de ces adolescents sont devenus des programmeurs professionnels. L'un d'eux a eu l'idée de créer le Raspberry Pi pour passer le flambeau.

La fondation Raspberry Pi a pour but d'aider les jeunes des années 2010 à emprunter la même voie, tout en y prenant du plaisir.

Tout cela permet de voir en quoi le Raspi est particulier. Son prix permet d'en acquérir un pour chaque projet. Et tu peux le laisser fonctionner sans cesse, car il consomme très peu. En plus, le système comporte dès le départ de nombreux outils pour aider à la création de logiciels, et tous sont gratuits.

La famille Raspberry Pi

En quelques années, le circuit original a fait la place à plusieurs générations successives (voir le Tableau 1.1). Il est indispensable de connaître les différences entre les circuits avant d'acheter.

Les premiers circuits s'appelaient tout simplement Raspberry Pi suivi de Model A ou Model B. Ils ont ensuite été remplacés par les modèles A+ et B+. En 2015, la fondation a lancé le Raspberry Pi 2 puis en 2016 le Pi 3.

La Figure 1.2 montre un modèle B+ et un modèle B.

Figure 1.2 : Deux modèles de Raspberry Pi des origines.

Toutes les générations, même la plus récente, conservent les mêmes dimensions. Le nombre de connecteurs a augmenté et les fonctions aussi.

À l'heure où tu lis ceci, le choix est simple : tu optes pour un Raspberry Pi 3 modèle B. C'est le plus récent. Le Pi 2 reste disponible, mais il coûte à peine moins cher.

Les modèles A et A+ sont des versions économiques dans lesquelles certains éléments ont été supprimés. Ce n'est pas la bonne solution pour démarrer.

Ce sont des variantes qui consomment moins et sont moins chères, ce qui les réserve à des projets totalement mis au point, par exemple dans le monde industriel. Pour bien profiter de ce livre, n'opte pas pour un modèle A ou A+.

Tableau 1.1 : Comparaison entre les modèles Raspberry Pi.

Modèle	*Domaine d'emploi*
A	Totalement obsolète et quasiment introuvable.
B	Totalement obsolète aussi.
A+	Plus lent qu'un Pi 2. Réservé aux projets spéciaux.
B+	Obsolète aussi.
Pi 2	Modèle B, envisageable.
Pi 3	Modèle B, c'est le modèle à acquérir.

Il n'existe actuellement pas de Modèle A ou A+ pour les deux dernières géné rations et il n'y en aura sans doute jamais.

Le plus récent modèle, le Pi 3, apporte un surcroît de puissance par rapport au Pi 2, mais surtout, il embarque dorénavant le Wifi et le Bluetooth ! Sur les précédentes versions, il fallait acquérir une carte ou un adaptateur.

Voici en résumé un bref comparatif entre le modèle Pi 2 B et le modèle Pi 3 B:

- Pi 2 Processeur à 900 MHz A7 avec 1 Go de mémoire 32 bits, vidéo à 250 MHz.
- Pi 3 Processeur à 1200 MHz A53 avec 1 Go de mémoire 64 bits, vidéo à 400 MHz.

Le seul point remarquable est que le Pi 3 consomme plus et chauffe donc plus. Il faut absolument lui fournir une alimentation suffisante et bien l'aérer. J'en parle plus loin.

Bien entourer son Raspi

Quand tu achètes le circuit Raspberry Pi, tu reçois un petit circuit électronique, et rien d'autre. Telle quelle, la carte ne sait rien faire. Tu peux la regarder, mais tu ne peux pas dialoguer avec elle. Il lui manque au minimum une alimentation, un écran, un clavier et une souris.

Ta liste d'équipement

Pour faire de cette carte un nano-ordinateur complet, il faut réunir quelques composants et les connecter. Cela constitue ton premier projet.

Voici la liste de ce qu'il faut prévoir :

- un adaptateur secteur ou alimentation ;
- un clavier USB ;
- une souris USB ;
- un écran ou un téléviseur ;
- une carte mémoire avec le système ;

- un câble réseau assez long ;
- plusieurs câbles.

Si tu ne disposes que d'un ancien modèle A ou B, il te faut également un concentrateur USB alimenté (USB hub).

Si possible, ne demande pas de l'aide à quelqu'un d'autre pour réunir les composants. Tu apprendras ainsi bien des choses sur l'informatique. Si tu es pressé, saute directement au tableau tout à la fin du chapitre qui permet de faire le point.

Concentrateur USB ou pas

Le concentrateur ou hub USB est une sorte de multiprise USB. Il en existe de deux sortes : les hubs passifs qui ne font que permettre plusieurs connexions, et les hubs alimentés, qui fournissent de l'énergie aux périphériques que tu y branches.

Tu n'auras besoin d'un concentrateur que si tu n'as pu trouver qu'une vieille carte A ou B originale, qui n'est plus en vente depuis longtemps. Sur ces modèles, le simple fait de brancher le clavier et la souris dans les deux ports USB pouvait entraîner des problèmes de tension.

Sur les versions récentes, tu disposes de quatre ports USB, ce qui permet de brancher tout ce dont tu as besoin. Sur les précédents modèles, il n'y avait que deux prises USB, et le fait de brancher le clavier et la souris n'en laissait plus aucune disponible. Dans ce cas, il n'est pas inutile de prévoir d'acquérir un concentrateur USB (toujours autoalimenté). La Figure 1.3 montre comment effectuer tous les branchements lorsque tu décides d'ajouter une multiprise USB.

Je rappelle qu'il faut prendre un modèle avec transformateur d'alimentation. Cela évite de puiser de l'énergie qui est normalement destinée à la carte Raspberry. L'inconvénient est que tu te retrouves avec un tas de fils encore plus gros qui traîne autour de la carte.

À partir des modèles A+ et B+, tu n'as plus besoin normalement d'un concentrateur (mais vois la remarque ci-dessus). La Figure 1.4 montre le branchement dans ce cas.

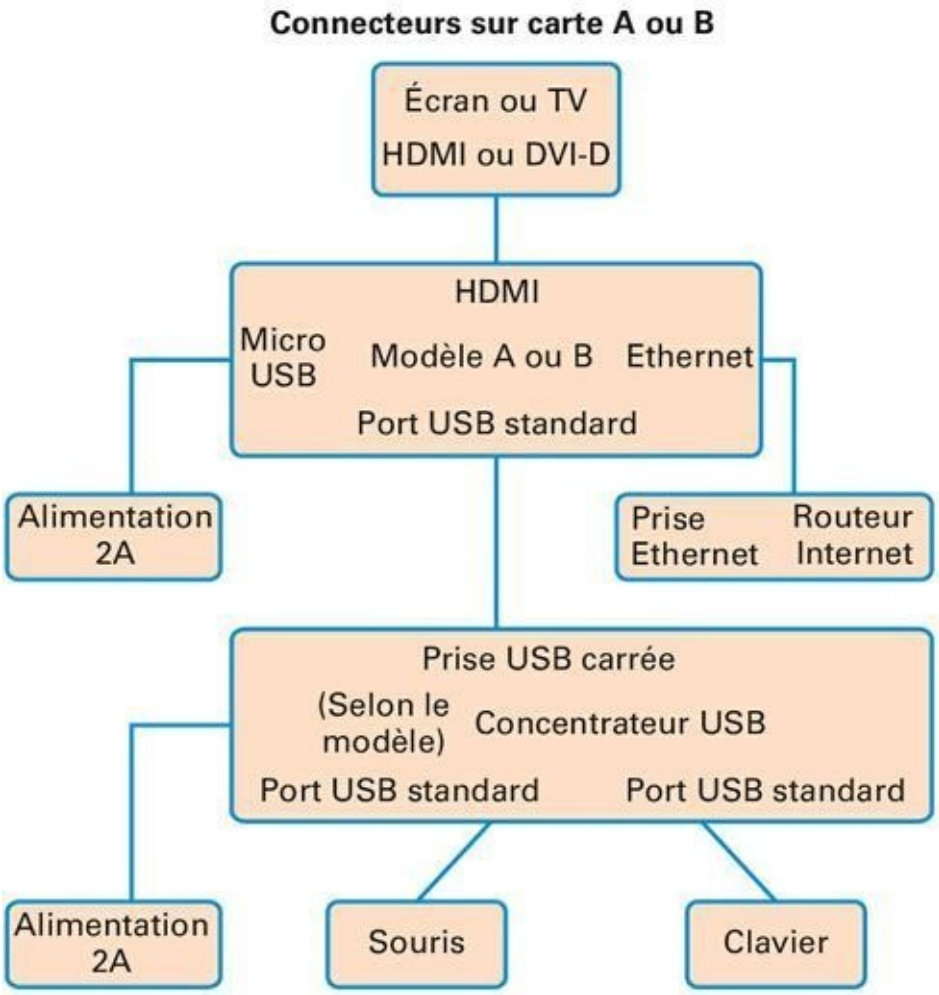


Figure 1.3 : Branchements d'un Raspberry avec concentrateur USB.

Le concentrateur est une boîte avec plusieurs connecteurs USB. Tu en branches un à la carte puis tu branches tous les périphériques aux autres ports. S'il possède son alimentation, il te garantit que chaque périphérique sera bien alimenté.

Si tu comptes brancher des équipements très gourmands, comme des moteurs de robotique ou des lasers, il faudra un concentrateur alimenté, même sur le plus récent modèle Pi 3. En revanche, un clavier ou une souris consomment très peu.

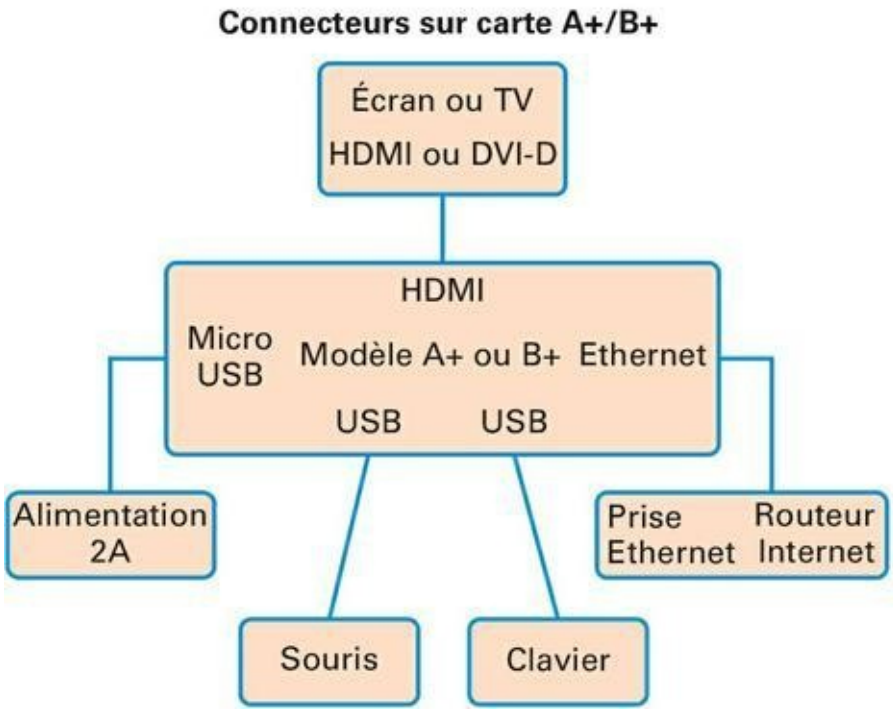


Figure 1.4 : Branchements d'un Raspberry sans concentrateur USB.

Choix de la souris et du clavier

N'importe quelle souris ou clavier USB conviendra. Tu peux te contenter d'un modèle filaire. Les modèles sans fil fonctionneront presque tous, à condition qu'ils aient un adaptateur USB. Tous les modèles de Logitech sont reconnus. En revanche, les souris et claviers Bluetooth risquent de ne pas fonctionner.

Ne dépense pas trop d'argent dans ces équipements. Les modèles de base sont parfaits.

Le Raspi n'offre pas assez de puissance pour envisager de jouer aux plus récents jeux vidéo. Inutile d'y brancher une souris de Gamer, du style Predator Ultra Galaxy Killer avec 15 boutons et un boîtier anguleux. Cela dit, si tu en as une de côté, tu peux t'en servir. Évidemment, les boutons en trop ne sont pas reconnus.

Choix de l'écran

Le Raspi peut fonctionner avec un écran ou avec un téléviseur.

Le plus simple consiste à brancher l'écran sur la prise HDMI. Les téléviseurs et les écrans actuels sont tous dotés d'une telle prise.

La Figure 1.5 montre où se trouve le connecteur HDMI, le deuxième en partant de la gauche.

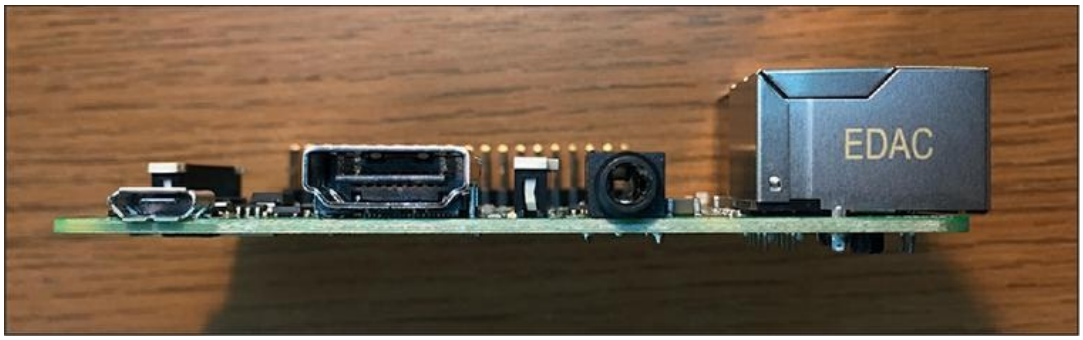


Figure 1.5 : Vue du connecteur HDMI.

Il n'est pas nécessaire d'avoir un écran dernier cri, car le Raspi produit de la vidéo de bonne qualité, mais pas de la HD. Du moment que l'écran est assez récent, il devrait convenir.

Certains écrans n'ont pas de prise HDMI, mais une prise DVI-D. Dans ce cas, procure-toi un adaptateur ou un câble HDMI d'un côté et DVI-D de l'autre. Tu en trouveras dans les magasins d'électronique. Dans ce cas, il n'y aura pas de son sur l'écran, car le signal DVI-D ne transporte que la vidéo. Il faudra alors brancher une enceinte sur la sortie jack.

Enfin, si ton écran n'a qu'une prise VGA, il te faut un adaptateur et un câble spécial. Cela dit, cela te reviendra peut-être moins cher d'acheter un écran plat plus récent d'occasion.

Sur les Raspi d'origine A ou B, il y a une grosse prise jaune qui permet de brancher une télévision analogique, ces fameux écrans cathodiques qui ont presque tous été envoyés au recyclage. Je te déconseille d'utiliser un tel écran, car l'image ne sera pas très nette ; tu auras du mal à lire les textes.

Le Raspi peut même fonctionner sans écran, c'est-à-dire contrôlé à distance depuis un autre ordinateur. Ce mode d'utilisation s'appelle

« sans tête » , *headless*. Dans ce cas, il n'y a besoin ni d'un écran, ni même d'une souris ou d'un clavier. Cela dit, même si le Raspi est configuré dès le départ pour permettre ce contrôle à distance, la configuration n'est pas simple. De plus, le télépilotage est différent sous macOS et sous Windows. Quand tu auras fini ce livre, si ce sujet t'intéresse, tu iras chercher des informations sur le Web en saisissant par exemple « headless Raspberry Pi » .

Les câbles et les connecteurs

USB, VGA, DVD, HDMI, que signifient tous ces sigles ? Tu trouveras facilement la réponse sur le Web.

Tu n'as pas besoin de savoir comment fonctionnent les communications par tous ces câbles. Tu n'as même pas besoin de savoir que HDMI signifie *High Definition Multimedia Interface*. Bon, maintenant, tu le sais.

En revanche, tu dois pouvoir distinguer les câbles pour savoir lequel il faut brancher et où. Tu te sers des photos de ce chapitre. La Figure 1.6 montre par exemple la prise réseau Ethernet à gauche et les quatre connecteurs USB d'une carte modèle B+.

Figure 1.6 : Vue du port Ethernet et des quatre ports USB.

Le monde de l'informatique regorge de sigles ou d'acronymes en trois ou quatre lettres, à tel point qu'en anglais on parle de TLA,

Three Letter Acronyms. Tu sais qu'un acrobate est quelqu'un qui grimpe tout en hauteur, et qu'une acropole est une ville en haut d'une colline. Un acronyme prend le « haut » des mots, en fait la première lettre de chaque.

Choix de la carte mémoire

Il n'y a pas de disque dur sur un Raspberry Pi. Aussi bien le système que les fichiers de données que tu vas créer sont stockés sur une carte mémoire. Les vieux modèles A et B utilisaient une carte de grand format avec une classe de performances entre 8 et 10. À partir des modèles A+ et B+, c'est une carte micro SD.

La Figure 1.7 montre le dessous d'une carte modèle B+. La carte mémoire est visible du côté droit.

Ta carte doit offrir au moins 4 Go d'espace, mais de nos jours, une carte de 8 Go ou 16 Go peut se trouver à un prix tout à fait raisonnable. Cela dit, inutile d'aller au-delà de 16 Go, sauf à vouloir stocker beaucoup de films.

La plupart des cartes microSD sont vendues dorénavant avec un adaptateur SD, ce qui te permettra d'utiliser le même système dans un vieux modèle A ou B et dans une carte récente.

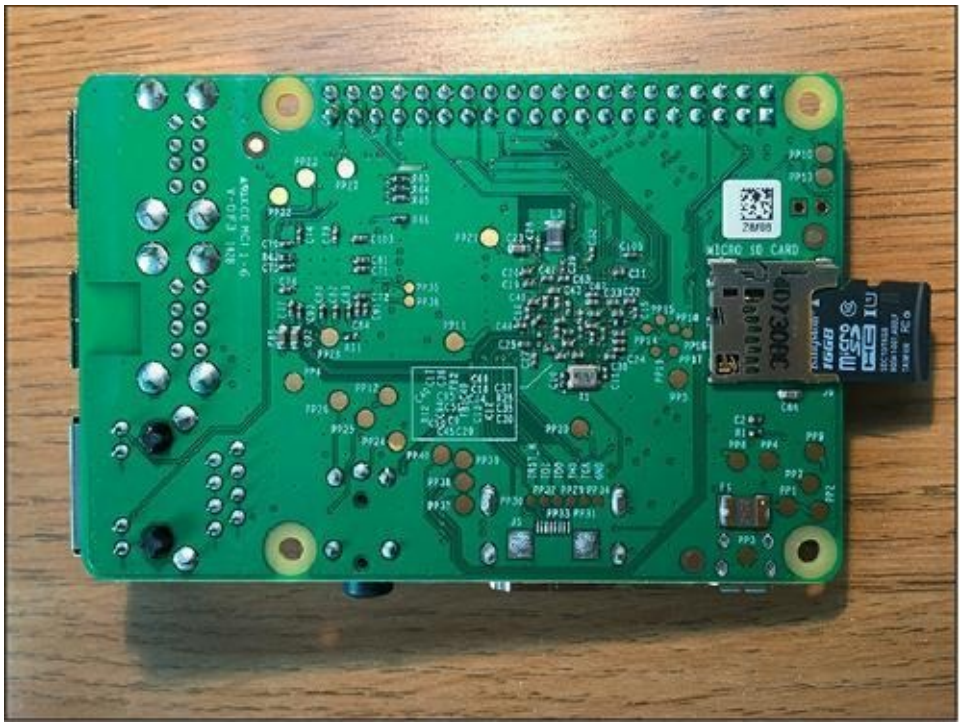


Figure 1.7 : Vue du dessous d'un Raspi avec la carte mémoire à droite.

Carte vide ou préparée ?

La solution la plus simple consiste à acheter une carte vide ou à réutiliser une carte. Dans ce cas, il faudra la formater et installer le système, comme je l'explique dans le chapitre suivant. Il faut bien sûr que ton ordinateur de bureau dispose d'un lecteur de cartes. Dans le cas contraire, tu en trouveras un pour environ cinq euros.

L'autre option consiste à acheter une carte sur laquelle est déjà copié l'installateur du système d'exploitation du Raspi. Cet outil porte le nom NOOBS. Tu trouveras ces cartes préparées chez les revendeurs de Raspberry Pi. La carte coûte un peu plus, mais elle te fait gagner du temps.

Choix de l'alimentation secteur

Le Raspi ne coûte pas cher, mais il ne faut pas négliger la qualité de son alimentation secteur, d'autant que le câble est spécifique. Sur la

carte, le connecteur d'alimentation est au format micro USB. La prise est suffisamment solide pour que le câble ne se retire pas par mégarde.

Pour tous les modèles, sauf le Pi 3 et ses successeurs, une alimentation de 2 ampères suffit. Puisque tu vas sans doute opter pour un modèle 3, prévois une alimentation secteur offrant 2,5 ampères.

Comme pour la carte mémoire, mieux vaut acheter l'alimentation auprès d'un revendeur agréé Raspberry Pi. N'oublie pas de vérifier qu'elle supporte 2,5 A si tu as besoin de ce modèle !

Tu trouveras des alimentations de 1 ou 1,5 A qui pourraient suffire à un ancien modèle A ou B, mais les choses iront en se dégradant au fur et à mesure que tu ajouteras des périphériques. Mieux vaut penser large dès le départ. Il existe des adaptateurs pour iPad 2 qui offrent 2,4 A, ce qui pourrait suffire si tu veux recycler le tien.

Le câble réseau Ethernet

Sauf si tu décides d'utiliser le mode sans fil avec le Raspberry Pi 3, il te faudra aussi un câble pour te brancher à ton routeur Ethernet. Ce câble sera à la norme Cat 5 ou Cat 6.

La mention Cat n'a rien à voir avec un chat anglais. C'est l'abréviation de catégorie (cinq ou six). Plus le chiffre est grand, meilleure est la qualité des conducteurs.

Si tu veux brancher plusieurs Raspberry sur un réseau local, il te faudra des câbles plus longs et un commutateur réseau (switch). Si cela fait partie de tes projets futurs, n'hésite pas à demander de l'aide autour de toi pour les branchements.

Autres équipements

Les composants que je présente maintenant ne sont pas obligatoires pour commencer tes aventures avec le Raspi. Ils n'en sont pas moins intéressants, et peuvent te faciliter la vie.

Un boîtier de protection

Le boîtier n’est pas indispensable, mais il protégera le circuit de la poussière, des objets qui tombent et des doigts des curieux.

Le Web regorge de boîtiers plus originaux les uns que les autres. L’objectif est toujours le même : protéger le circuit. Note que le format des boîtiers n’est pas le même pour les anciens modèles A/B et les plus récents. La Figure 1.8 montre le boîtier transparent, très apprécié.

Si tu vis dans un pays chaud, pense à choisir un boîtier qui possède des trous d’aération afin d’évacuer la chaleur, surtout sur le Pi 3.

Si tu n’as pas de boîtier, prend toujours soin de poser le Raspi sur quelque chose qui ne conduit pas l’électricité. Un bout de carton, ou une plaque en plastique, même une assiette, fera l’affaire. Évite tout plateau métallique, qui pourrait provoquer un court-circuit fatal.

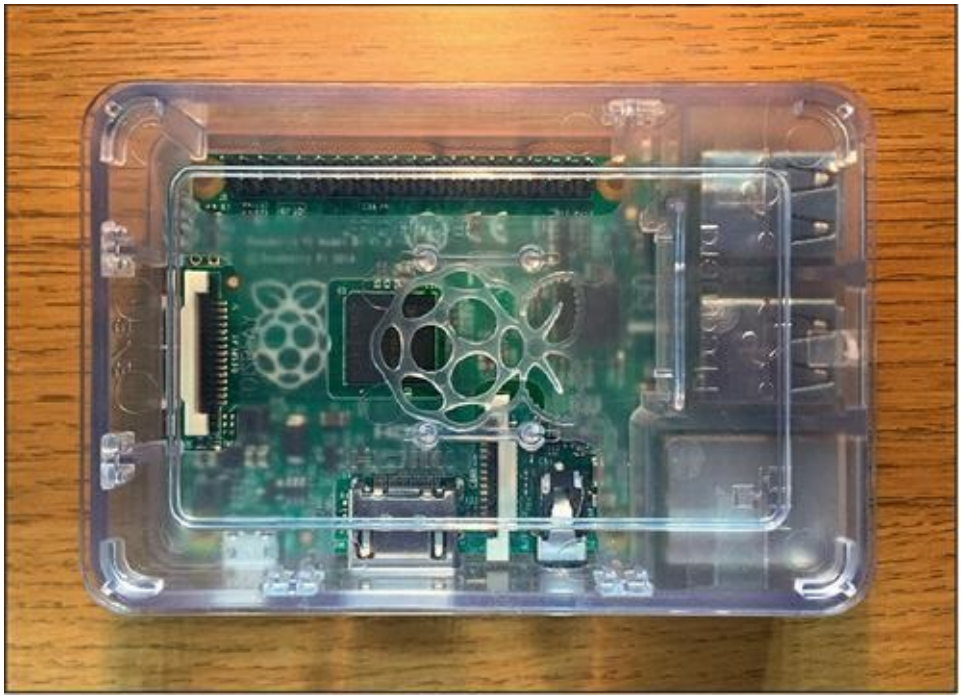


Figure 1.8 : Boîtier de protection transparent.

En cas de court-circuit, si tu as de la chance, le circuit s'arrête et tu peux le redémarrer. Si tu as moins de chance, le circuit sera détruit. Il ne te restera plus qu'à casser ta tirelire pour en acheter un autre.

Le Raspi déteste l'électricité statique. Avant de le prendre dans une main, arrange-toi pour bien le prendre par les côtés ou bien par les prises USB ou réseau. Ne touche jamais les composants soudés avec les doigts. Ne pose pas non plus le circuit sur un tapis synthétique, objet bien connu pour générer de l'électricité statique. Enfin, évite toute projection de liquide, comme tu t'en doutes.

Si tu ne veux pas dépenser de l'argent pour le boîtier, tu peux récupérer sur le site officiel Raspberry un modèle à imprimer et à découper en carton. Va sur le site www.Raspberry.org et cherche les mots « card case » ou « punnet ».

Le Wifi

Si tu as opté pour le Pi 3, tu n'as pas besoin de lire cette section, car il est doté du Wifi et du Bluetooth en standard. Pour un modèle plus ancien, il te faut ajouter un adaptateur Wifi USB.

Tous les adaptateurs Wifi ne sont pas reconnus par le Raspi. Certains fonctionnent au départ puis s'arrêtent sans raison. D'autres fonctionnent très lentement. Dans la mesure du possible, utilise une connexion filaire avec un câble Ethernet, sauf sur les modèles les plus récents dotés du Wifi.

Ajouter une caméra

La caméra est un périphérique souvent utilisé avec un Raspi. La plupart des modèles USB fonctionneront bien, mais il faut éviter les plus anciens et les trop économiques à moins de dix euros.

Une solution sûre est le module `Camera` officiel de Raspberry. C'est un petit bijou d'électronique mesurant 2 cm de côté qui offre des performances étonnantes. Et le Raspi est doté dès le départ du logiciel approprié, alors qu'avec une autre caméra USB, il reste à l'écrire ou du moins à le télécharger puis à l'installer.

Mais le module caméra est délicat. Pour une utilisation régulière, mieux vaut le stabiliser dans un boîtier. Il existe même des boîtiers pour caméra Raspberry sur le Web.

La Pi Camera existe en deux variantes : standard et sans filtre infrarouge (noIR). La variante noIR permet de prendre des photos et vidéos de nuit. La plupart des ventes concernent la variante standard, l'autre étant plutôt destinée à l'observation d'animaux nocturnes ou à la chasse aux fantômes ou aux feux-follets.

Une enceinte ou un casque

Le Raspi comporte une sortie jack audio standard ; tu peux donc y brancher un casque ou un câble jack stéréo comme ceux des telliphones et des baladeurs MP3. Le plus simple est d'y brancher une enceinte nomade comme il en fourmille de nos jours. Ce sera d'ailleurs indispensable si ton écran n'a pas d'enceintes intégrées. Dans ce cas, même si le son est transmis *via* le câble HDMI, tu ne pourras par l'entendre.

Sauf le modèle Pi 3, il n'y a pas de Bluetooth sur le Raspi. D'ailleurs, mieux vaut utiliser la sortie audio filaire. Le circuit n'est pas compatible avec les stations d'accueil Apple et Android.

Un achat simplifié

La solution facile pour démarrer consiste à acheter un Pi Starter Kit qui réunit tous les composants requis, sauf évidemment l'écran et son câble. Tu trouveras aisément en cherchant « Raspberry Pi Starter Kit » .

On trouve encore des kits de démarrage qui contiennent un Model B au lieu du B+. Ne confonds pas kit de démarrage et kit électronique. Ce dernier te sera utile plus tard seulement. Il faut que le kit contienne au moins l'alimentation, le clavier, la souris, la carte mémoire et les câbles.

Si tu as déjà un clavier et une souris, cherche le kit de base : une carte Pi, une carte mémoire et une alimentation. C'est moins cher qu'un kit

complet.

Résultat des courses

Si tu préfères ne pas opter pour un kit carte + accessoires, vois chez toi s'il n'y a pas des périphériques devenus inutiles. Fouille dans les placards au cas où il resterait un clavier ou une souris, des câbles ou même un écran.

Si tu n'as rien trouvé, demande aux amis ou cherche d'occasion sur Internet.

Le Tableau 1.2 propose une liste d'approvisionnement pour ton système Raspi. Il suffit de cocher la colonne appropriée selon que tu possèdes déjà ou pas l'équipement nécessaire.

Tableau 1.2 : Liste de courses pour le Raspi.

Composant Requis pour	*J'en ai un*	*Je dois en acheter un*
Écran ou TV	Tous	
Clavier USB	Tous	
Souris USB	Tous	
Alimentation 2 A	Tous sauf Pi 3	
Alimentation 2,5 A	Pi 3	
Carte Micro SDHC	Tous sauf A/B	
Carte SDHC	Qu'anciens A/B	
Lecteur de carte	Si pas sur PC ou Mac	

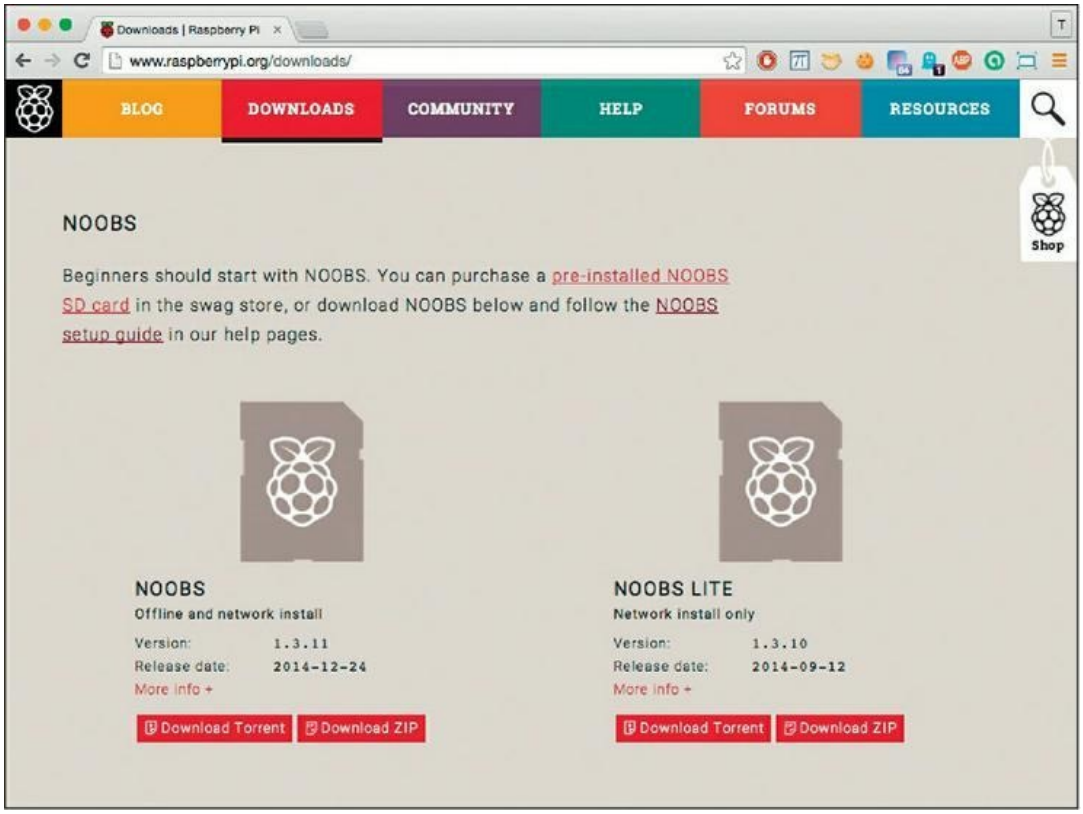
Câble Ethernet	Tous	
Hub USB	Tous en option	
Alim Hub USB	Tous en option	
Clé WiFi	Sauf Pi 3	
Câble HDMI/HDMI	Tous	
Câble HDMI/DVI	Si sans HDMI	
Boîtier	Tous en option	
Caméra	Tous en option	

Chapitre 2
Un cerveau pour ton Raspi

Tant qu'elle n'est pas animée par un logiciel qui s'appelle le système d'exploitation, la carte Raspberry Pi n'est qu'un morceau plastique avec des composants soudés dessus.

Le système est le cerveau de ta machine. Comparé au cerveau d'une fourmi, il est ridiculement simple, mais il est indispensable pour que tu puisses profiter de ton Raspi.

Dans ce chapitre, nous allons voir pas à pas comment mettre en place ce cerveau sur la carte mémoire.



Les systèmes d'exploitation

En informatique, tout ordinateur possède un cerveau qui est son système d'exploitation, en anglais OS pour *Operating System*.

Ce système est le chef d'orchestre de l'ordinateur. C'est lui qui contrôle l'utilisation de toutes les ressources. Lorsque tu demandes à la machine de réaliser une action, c'est le système qui réceptionne ta demande, lance l'action et montre le résultat.

Le système comporte notamment des sous-programmes pour gérer les périphériques principaux que sont le clavier, la souris, l'écran et le périphérique de stockage des données (ici, c'est une carte mémoire). C'est aussi le système qui se charge de la connexion Internet et des échanges de données.

Il existe plusieurs systèmes d'exploitation principaux que présente le Tableau 2.1.

Tableau 2.1 : Quelques systèmes d'exploitation répandus.

Matériel	*Système*
PC	Windows
PC	Linux
mac	OS
iPhone	iOS
Autres telliphones	Androïd
Raspberry Pi	Linux (voir dans ce chapitre)

Un logiciel prévu pour un système ne fonctionnera pas avec un autre. Si tu as un jeu vidéo pour Mac, tu ne peux y jouer sous Windows. De même, un mini-jeu prévu pour iOS ne fonctionnera ni sous macOS, ni sous Windows.

Les programmeurs prévoient de plus en plus souvent plusieurs versions de leur projet pour s'adapter à plusieurs systèmes ou plates-formes. Le résultat est un logiciel multiplateforme. Le supplément d'efforts que cela entraîne ne rend cette solution intéressante que pour les logiciels à grande diffusion.

Linux et le Raspberry Pi

Un des premiers objectifs des créateurs du Raspberry Pi a été de maintenir un prix bas. Cela a immédiatement mis hors de course les systèmes propriétaires qui réclament de payer une licence pour chaque exemplaire. Tu ne peux donc pas utiliser ton Raspi ni sous Windows, ni sous macOS.

Il existe une version d'Androïd pour le Raspi, mais elle ne fonctionne pas encore très bien. Je n'en parlerai donc pas.

Pour animer le Raspi, tu vas utiliser un système d'exploitation libre et gratuit qui s'appelle Linux. Au départ, Linux n'est pas vraiment conçu pour les débutants, mais plutôt pour les passionnés d'informatique. Dans Linux, tout peut être modifié et adapté à tes désirs.

La société Microsoft a annoncé qu'il y aurait un jour une version gratuite de Windows 10 pour le Raspberry Pi 2, une version allégée pour les projets d'électronique. Ce n'est pas la version de Windows que tout le monde connaît avec sa suite bureautique et les autres applications habituelles. Pour le moment, considère que le projet Windows pour Raspi n'a aucun rapport avec le système Windows.

Linux est accompagné d'une foule d'outils qui vont te guider dans tes premiers pas si tu veux devenir programmeur. Tous ces outils sont gratuits et installés dès le départ. Et cela sans compter les nombreux autres outils disponibles, toujours gratuits.

Puisque Linux permet une personnalisation maximale, nombreux sont ceux qui en ont fait une version personnelle. Les meilleures personnalisations sont devenues des distributions. Il existe par exemple une distribution Linux consacrée au multimédia, avec de nombreux outils audio et vidéo, une autre destinée à l'éducation, de la maternelle à l'université, d'autres encore pour le monde scientifique.

Sache que modifier Linux n'est pas simple, même pour un expert. Il faut accumuler suffisamment d'expérience avant de toucher à quoi que ce soit. C'est en tout cas totalement différent de l'approche Windows ou macOS dans lesquels tu ne peux quasiment rien changer, à part en surface. Les propriétaires de ces systèmes ne t'autorisent pas à les modifier.

Le système RaspBian

Le système d'exploitation prévu pour ton nano-ordinateur porte le nom Raspbian. C'est une version spécialement personnalisée pour le Raspberry Pi de l'une des distributions les plus répandues de Linux qui porte le nom Debian.

Le bureau graphique de Raspbian (Figure 2.1) a un air de famille avec les précédentes versions de Windows, avant l'apparition des carreaux de mosaïque dans Windows 8.

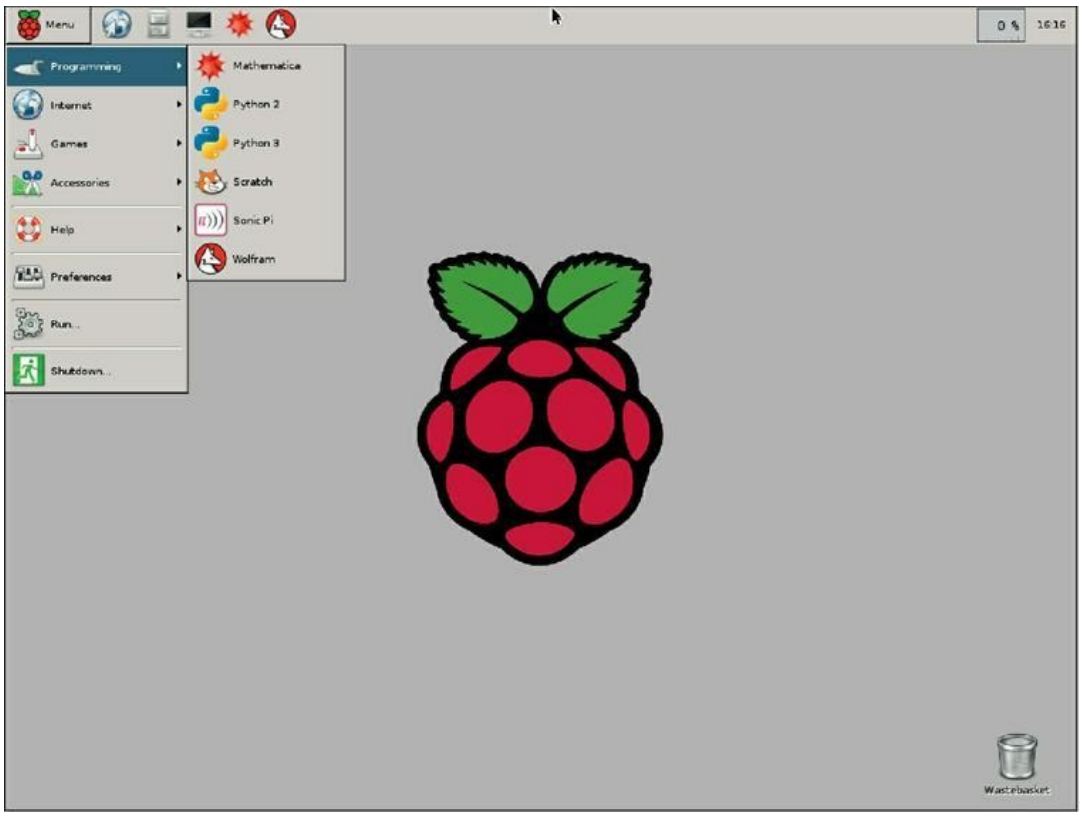


Figure 2.1 : Vue générale du bureau de RaspBian.

Si tu as choisi d'acheter une carte mémoire avec le système préinstallé et que tu constates au démarrage que le fond du bureau est blanc au lieu de gris, c'est que tu as une ancienne version. Quasiment toute la suite du livre restera utilisable, mais certains menus et options ne seront pas au même endroit. Le mieux est de reformater la carte et d'installer la version la plus récente de NOOBS, comme je l'explique dans les pages suivantes.

Le bureau de RaspBian a également un air de famille avec le bureau du système macOS du Mac, mais avec un peu moins d'effets artistiques dans les textes.

Dans tous les cas, si tu sais utiliser l'interface d'un Mac ou celle de Windows, tu ne seras pas dépaysé avec RaspBian. Je présente les

différences dans le Chapitre 5. Tu n'auras pas à découvrir beaucoup de choses différentes.

Le nom Debian provient des prénoms des deux créateurs de cette distribution Linux, Deb et Ian.

La version Debian de Linux est très répandue parce qu'extrêmement polyvalente. Elle ne propose pas les plus récentes évolutions du système Linux, mais cela lui permet d'être beaucoup plus stable que les versions réservées aux passionnés.

Debian réunit la plupart des outils disponibles sous Linux. La version pour le Raspi y ajoute quelques jeux.

Le préparateur NOOBS

Pour faciliter la création de ta carte système Raspbian, tu vas utiliser un outil appelé NOOBS. C'est l'acronyme anglais de *New Out Of the Box Software* (logiciel neuf sortant du carton). Ce préparateur va configurer le système sur la carte. Voici les étapes de l'utilisation de ce préparateur.

1. Tu dois d'abord installer NOOBS sur une carte mémoire ou acheter une carte qui le contient déjà.

   La procédure est expliquée en détail aussi bien pour macOS que pour Windows dans les pages suivantes.

2. Tu insères la carte mémoire dans le Raspi éteint.
3. Tu allumes le Raspi.
4. Tu suis les instructions qui s'affichent pour installer le système Raspbian.
5. Tu redémarres le Raspi.

6. À partir de ce moment, tu utilises le système RaspBian.

Une fois que RaspBian est installé, tu ne verras plus apparaître le préparateur NOOBS. Il est toujours présent sur la carte mémoire, mais le Raspi va l'ignorer et démarrer directement dans le système RaspBian.

Il existe une combinaison de touches spéciale qui permet d'accéder à nou veau au préparateur NOOBS, mais tu n'en auras pas besoin.

Utiliser NOOBS préinstallé

Si tu as opté pour un kit de démarrage Raspberry avec une carte mémoire, elle est normalement préparée avec un système. Tu peux également acheter indépendamment une carte préformatée avec NOOBS.

À partir du moment où tu disposes d'une telle carte, tu n'as plus besoin de lire la suite de ce chapitre.

Attention au format de la carte. Sur tous les modèles récents de Raspi, il te faut une carte au format micro SD. Si tu réutilises un ancien modèle A ou B, il te faut une carte mémoire SD à l'ancien grand format.

Créer sa carte NOOBS

Au lieu d'acheter une carte NOOBS prête à l'emploi, tu peux choisir de la créer toi-même. C'est d'ailleurs conseillé pour des raisons de pédagogie.

Je dois te prévenir que la préparation de la carte mémoire représente un certain travail. Il faut télécharger un outil de formatage, télécharger l'image de NOOBS, la désarchiver puis la transférer sur la carte mémoire. Cela prend un certain temps.

Voici les principaux ingrédients pour la création d'une carte :

- un PC sous Windows ou un Mac ;
- une carte mémoire vide ;
- un lecteur de carte mémoire interne à l'ordinateur ou externe ;
- plusieurs logiciels disponibles gratuitement sur le Web ;
- entre une et deux heures de temps.

La carte mémoire

J'ai déjà expliqué dans le chapitre précédent quel genre de carte il te faut. Si possible, cherche la carte la plus rapide possible, par exemple une classe 10. La Figure 2.2 montre une carte grand format des premiers modèles et la carte micro SD des modèles actuels avec un adaptateur entre les deux.

Figure 2.2 : Les deux modèles de cartes mémoire : SD HC et micro SDHC.

Réservation d'un PC sous

Windows ou d’un Mac

Cette étape est facile. N’importe quel Mac ou PC sous Windows ayant moins de cinq ans suffira. L’essentiel est que la machine possède un lecteur de carte.

Certains modèles de Mac et de PC n’ont pas de lecteur de carte interne. Dans ce cas, il te suffit de te procurer un lecteur externe se branchant sur un port USB. J’en ai déjà parlé dans le chapitre précédent.

Le lecteur externe de la Figure 2.3 est assez volumineux. Tu en trouveras des beaucoup plus compacts.

Figure 2.3 : Exemple de lecteur de cartes mémoire externe.

Vérifie bien que le lecteur est capable de lire des cartes au format micro SD.

Tu peux insérer ta carte mémoire dans le lecteur alors qu’elle est encore vide. Elle devrait apparaître en tant que disque externe dans le

Gestionnaire de fichiers de Windows ou le Finder du Mac.

Pour l'instant, la carte est vide ou presque. Il va falloir la préparer.

Les étapes de création d'une carte NOOBS

La procédure est la même sous macOS et sous Windows, avec seulement quelques détails différents. Voici cette procédure :

1. Téléchargement puis installation de l'outil de formatage de carte gratuit (il te faut ton mot de passe système !).
2. Formatage de la carte mémoire avec l'outil.
3. Téléchargement de l'image du préparateur NOOBS et du système.
4. Extraction du contenu de l'archive compressée de NOOBS dans un dossier.
5. Copie du contenu du dossier vers la carte mémoire.
6. Éjection de la carte.
7. Extraction physique de la carte du lecteur.

À partir de ce moment, la carte est prête à servir sur ton Raspi. Lors du prochain démarrage, cela lancera NOOBS qui installera Raspbian.

Le gros avantage de NOOBS est de simplifier l'installation du système. Certains revendeurs proposent à la vente des cartes mémoire sur lesquelles le système Raspbian est déjà installé et presque configuré. Mais en suivant la procédure des pages suivantes, tu seras beaucoup plus autonome par la suite.

Tu peux utiliser les outils standard de Windows ou de macOS pour formater la carte, mais cela ne fonctionne pas toujours. Voilà

pourquoi il vaut mieux récupérer l'outil officiel disponible sur le site de la fondation *sdcard.org*.

Création de la carte NOOBS sous macOS

Voyons d'abord comment procéder sous macOS. La procédure sous Windows est très similaire.

Téléchargement de l'outil de formatage

Tu n'as besoin de télécharger l'outil qu'une fois. Tu peux ensuite préparer autant de cartes système que nécessaire.

1. Ouvre ton navigateur Web et rends-toi dans la page suivante :

   `www.sdcard.org`

2. Dans le bandeau des menus, choisis `Downloads` puis cherche à gauche les boutons de téléchargement (Figure 2.4).

3. Clique le bouton `Download SD Formatter for Mac`.

4. Fais défiler la page des conditions juridiques pour pouvoir cliquer en bas le bouton `Accept`.

   Le téléchargement du fichier *.pkg* sur ton Mac s'engage. Tu devras peut-être choisir où le fichier doit être déposé.

5. Double-clique le fichier que tu viens de télécharger et suis les instructions qui apparaissent.
6. Affiche le contenu du dossier `/Applications` et cherche le nom de l'application **SDFormatter**.

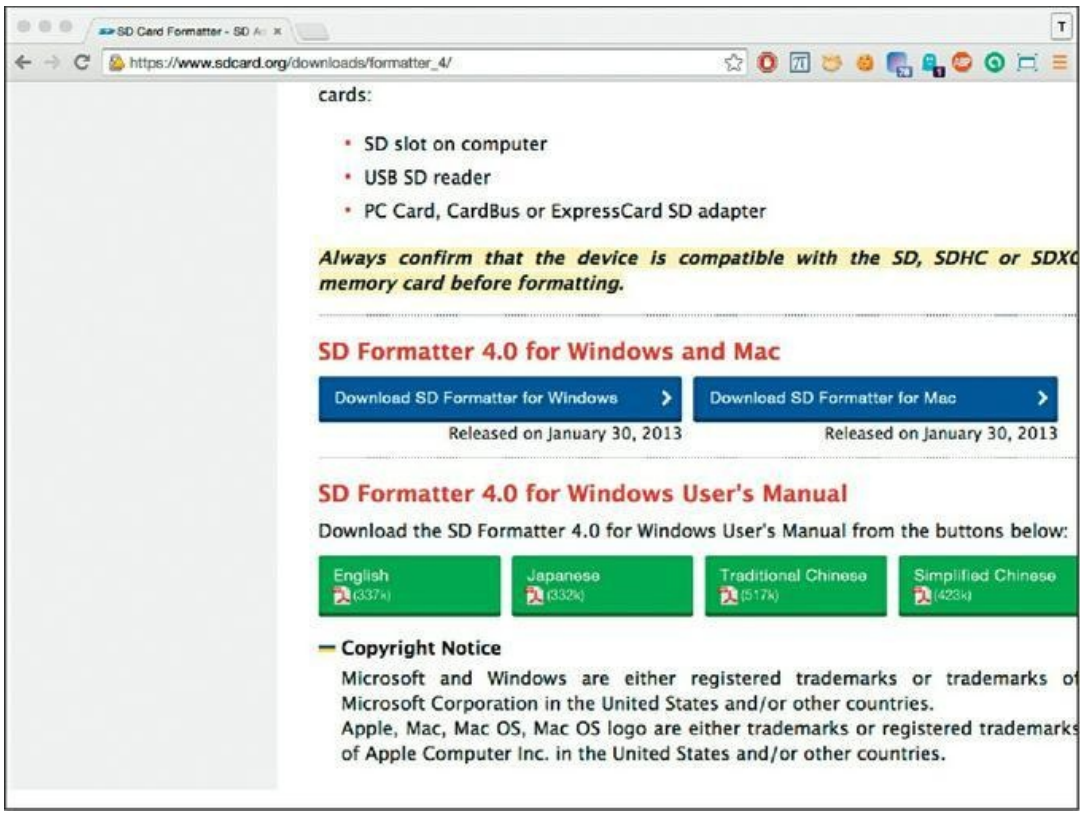


Figure 2.4 : La page de téléchargement du formateur.

Formatage de la carte sous macOS

Cette opération est très simple. Il suffit de régler quelques options :

1. Si tu utilises un lecteur externe, branche-le à un port USB puis insère la carte. Si le Mac possède un lecteur interne, insère la carte directement.

2. Double-clique l'outil `SDFormatter.app`.

3. Saisis ton mot de passe système pour autoriser l'installation et confirme par OK.

4. Lorsque tu vois apparaître la fenêtre de l'outil (Figure 2.5), ouvre la liste du haut et sélectionne ta carte mémoire.

Cette sélection n'est nécessaire que s'il y a plusieurs cartes.

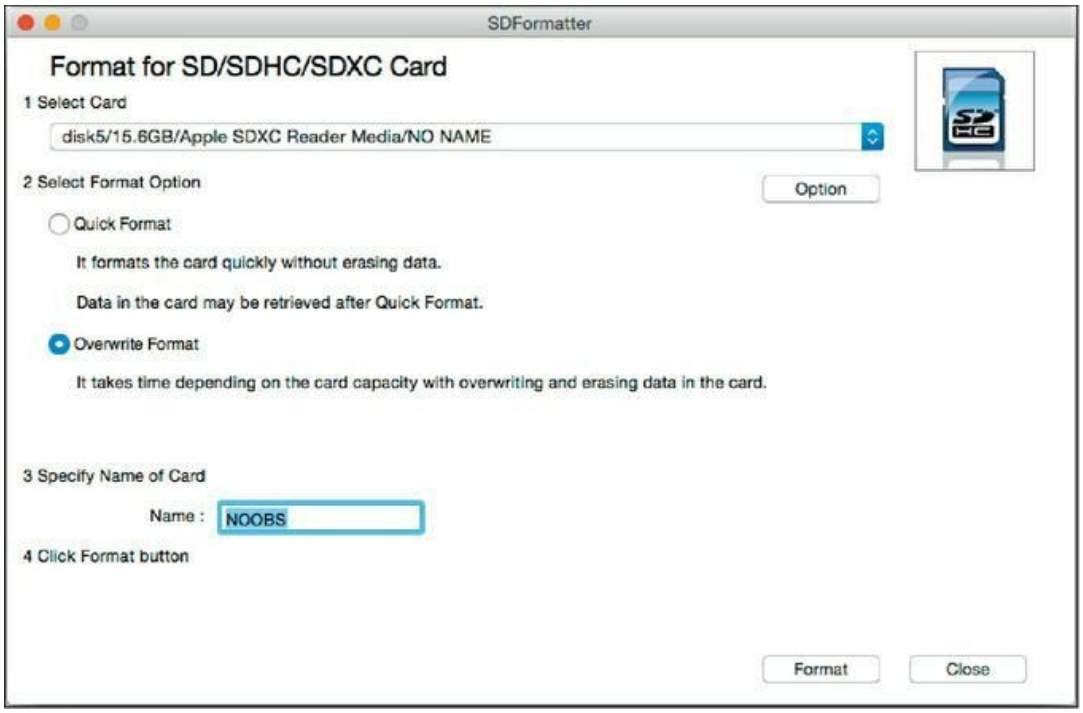


Figure 2.5 : Page de préparation du formateur de carte.

Si tu ne vois apparaître aucun nom de carte dans le menu, vérifie qu'elle est bien insérée. Inversement, s'il y a plusieurs cartes, il faut t'assurer que tu as bien sélectionné celle dont tu veux effacer le contenu.

5. Dans la zone de texte Name, saisis NOOBS.

Ce n’est qu’une option, car le nom de la carte n’a aucune importance. Si tu as plusieurs cartes, cela te permettra de savoir ce que tu as mis sur celle-ci.

6. Clique le bouton `Options` dans la page puis choisis `Yes` pour l’option `Logic Address Adjustement`.

7. Clique OK pour refermer la boîte d’options.

8. Dans la boîte principale, clique l’option de formatage `Overwrite format`.

Cette option est un bouton radio, c’est-à-dire que lorsque tu la sélectionnes, les autres boutons radio du même groupe se désélectionnent d’office, comme pour changer de gamme d’ondes sur les anciens récepteurs radio.

9. Après avoir tout vérifié, clique le bouton `Format` et arme-toi de patience.

L’opération de formatage peut durer de dix minutes à une demi-heure selon le volume de la carte. Heureusement, il y a une jauge de progression, ce qui te permet de savoir si tu as le temps d’aller te promener un peu.

10. Dès que le formateur a terminé son travail, clique le bouton `Close`.

Dorénavant, ta carte mémoire a été reformatée de telle sorte que le Raspi puisse l’utiliser. Il ne reste plus qu’à y implanter le contenu.

Téléchargement de NOOBS

Le préparateur NOOBS est bien sûr disponible sur le site officiel de la fondation Raspberry Pi (Figure 2.6).

1. Accède à la page suivante :

www.raspberrypi.org

2. Dans la page d'accueil, clique en haut le bouton DOWNLOADS pour accéder à la page des téléchargements.

   Tu dois trouver une icône pour NOOBS.

3. Clique l'icône NOOBS.

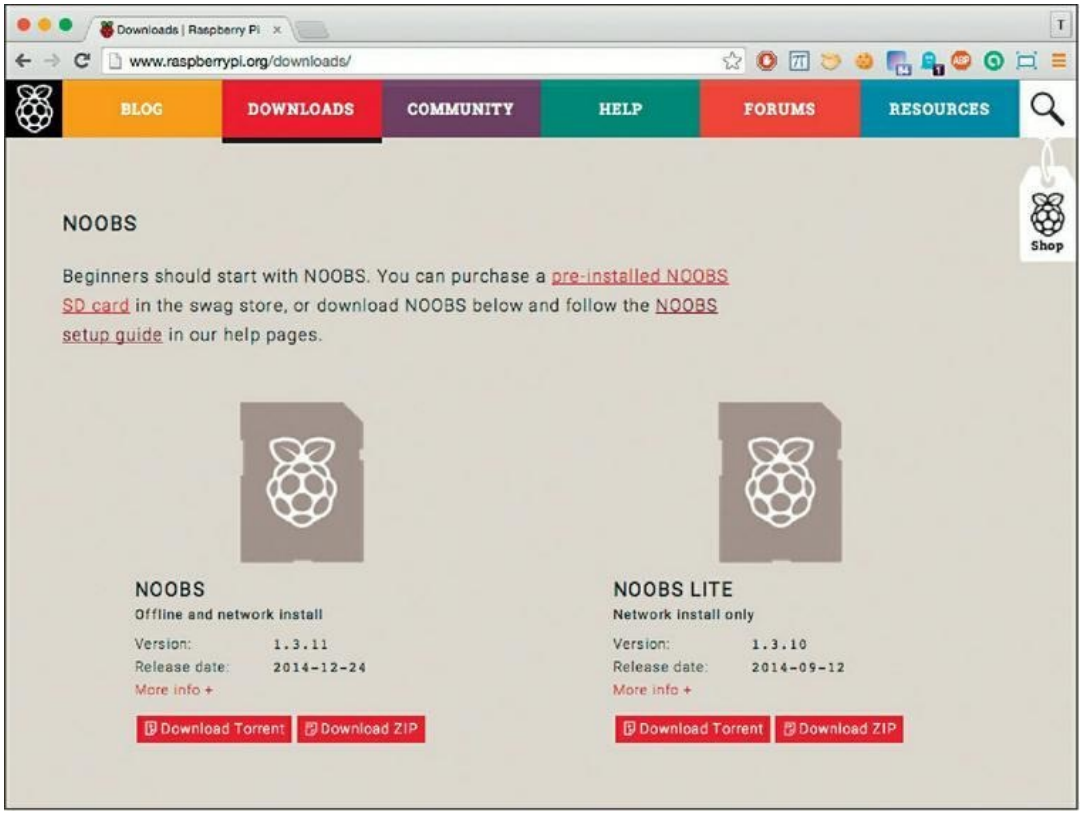


Figure 2.6 : La page de téléchargement de NOOBS.

Tu as le choix entre deux variantes de NOOBS : la variante standard qui contient la totalité du système Raspbian, et la variante Light qui a besoin de télécharger l'essentiel du système depuis Internet pour l'installer. La variante standard est à privilégier, car rien ne dit que ton Raspi va détecter une connexion Internet dès sa mise sous tension.

Choisis donc de télécharger la version standard au format compressé ZIP avec le bouton `Download ZIP` de la version `standard`. Prends note de l'endroit où est stocké le fichier téléchargé.

Par défaut, il est placé dans le dossier */Téléchargements* de ton dossier utilisateur. Il est possible cependant que le navigateur le stocke ailleurs, s'il a été configuré autrement. Il est également possible qu'il te demande où tu veux le stocker. Dans tous les cas, sois bien attentif au moment de choisir où le fichier va être déposé, afin de le retrouver facilement par la suite.

Le fichier d'archive NOOBS est assez volumineux (presque 800 Mo). Autrement dit, son téléchargement va prendre un certain temps, surtout si ta connexion Internet est un peu poussive. Si tu n'es pas seul à utiliser cette connexion, préviens ton entourage.

Extraction du contenu de NOOBS

Pour faire apparaître le contenu de l'archive, il suffit de double-cliquer le nom du fichier que tu viens de télécharger au format ZIP. Le Finder va créer un dossier contenant toute l'arborescence qu'il a trouvée dans l'archive.

La Figure 2.7 montre le résultat du désarchivage. Le fichier archive est encore visible dans la liste.

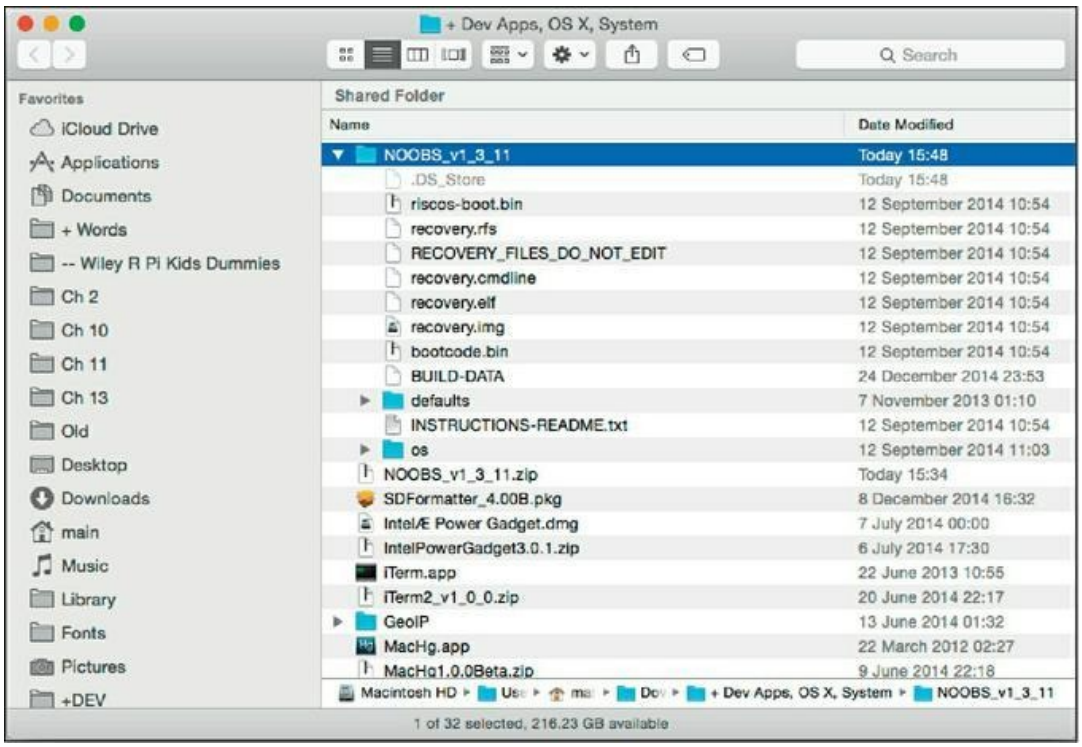


Figure 2.7 : Résultat de la décompression de l'archive NOOBS.

Transfert des fichiers sur la carte

1. Pour copier facilement tous ces fichiers sur la carte, ouvre une autre fenêtre du Finder et navigue jusqu'à la carte mémoire qui doit être disponible dans la liste des périphériques dans le bas de la fenêtre.
2. Dans l'autre fenêtre, sélectionne tous les fichiers du premier niveau du dossier NOOBS.
3. Avec un glisser-déposer, copie l'ensemble des fichiers vers l'autre fenêtre Finder, celle qui montre la carte vide (Figure 2.8).

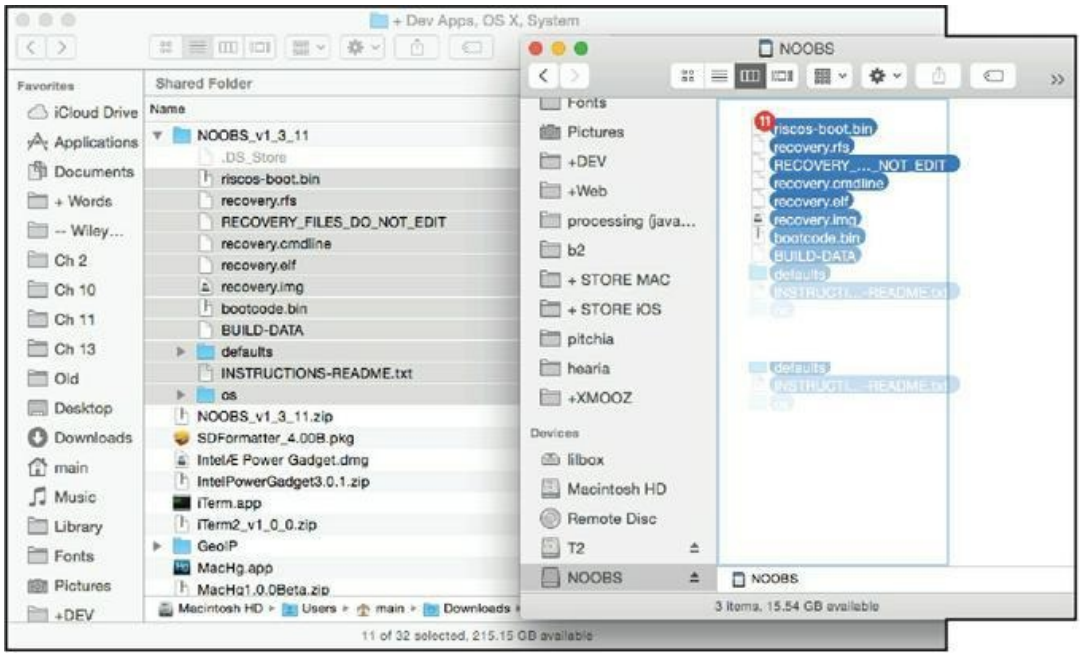


Figure 2.8 : Transfert des fichiers vers la carte.

4. Dès que tu relâches le bouton de souris, le Finder transfère les fichiers. Une fois que l'opération est terminée, tu peux éjecter la carte et l'extraire du Mac ou du lecteur. La carte est prête à être insérée dans le Raspi. Mais ne va pas trop vite en besogne, nous verrons ça dans un chapitre ultérieur.

Attention ! Il ne faut pas copier le dossier NOOBS, mais les fichiers que contient le dossier. Lorsque le Raspi va démarrer, il ne saura pas descendre dans un dossier portant le nom NOOBS qu'il trouverait sur la carte. Il doit trouver les fichiers du premier niveau dans la racine de la carte.

Création de la carte NOOBS sous Windows

Comme annoncé, la procédure sous Windows est très similaire à celle sous macOS.

Téléchargement de l’outil de formatage

Je ne vais citer que les différences dans la procédure par rapport à celle pour macOS.

1. Dans ton navigateur Web, rends-toi à la page suivante :

   `www.sdcard.org`

2. Dans le bandeau des menus, choisis `Downloads` puis cherche à gauche les boutons de téléchargement (Figure 2.9).

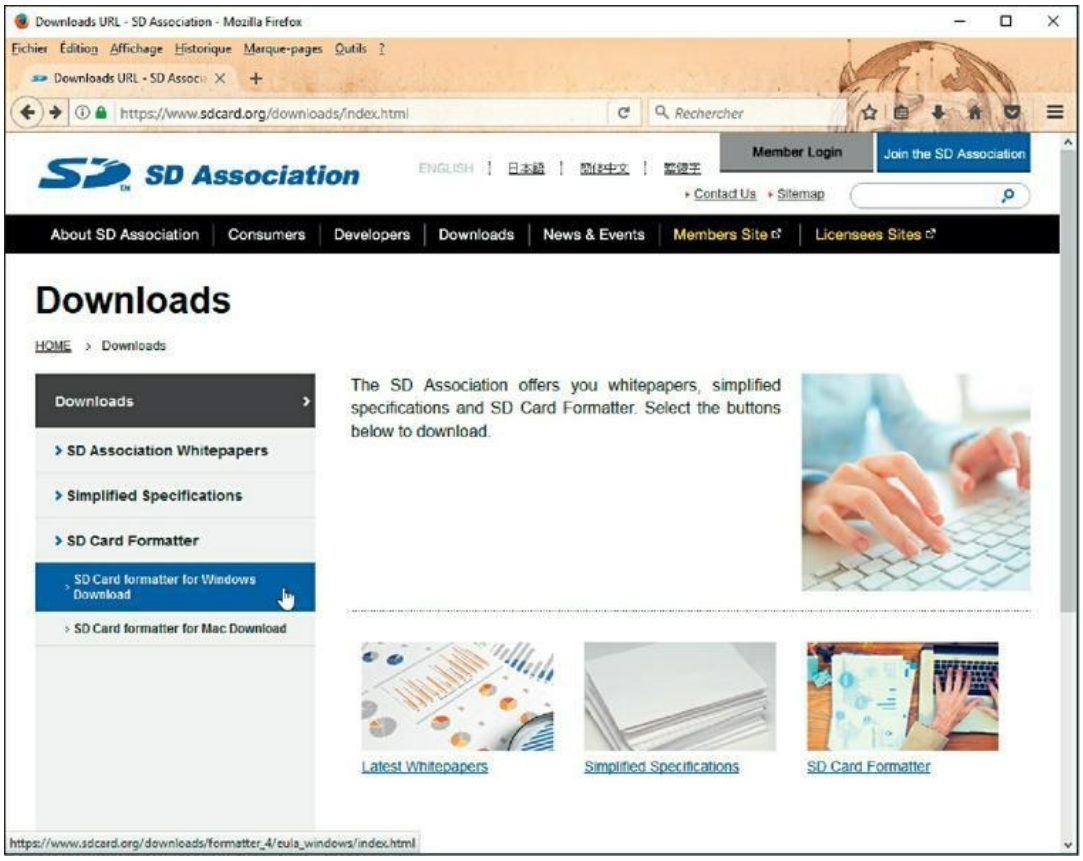


Figure 2.9 : La page de téléchargement du formateur.

3. Clique le bouton `Download SD Formatter for Windows`.
4. Fais défiler la page des conditions juridiques pour pouvoir cliquer le bouton `Accept`.
5. Surveille le téléchargement du fichier compressé vers ton PC.
6. Double-clique le nom du fichier téléchargé.
7. Dès que tu vois apparaître le nom *setup.exe* dans le Gestionnaire de fichiers, double-clique pour lancer l'installateur.

8. Suis les instructions pour installer l'outil de formatage.

Une fois que c'est terminé, une nouvelle icône pour le formateur doit être apparue sur le bureau.

Formatage de la carte mémoire sous Windows

La version Windows de l'outil offre les mêmes options que la version Mac, mais la boîte n'a pas exactement le même aspect.

Voici comment formater la carte sous Windows :

1. Insère la carte mémoire dans le lecteur externe ou dans le lecteur interne du PC.
2. Double-clique l'icône `SDFormatter` pour démarrer l'outil.
3. Tu vois apparaître la fenêtre montrée en Figure 2.10. Commence par sélectionner la carte mémoire dans le menu `Drive`.

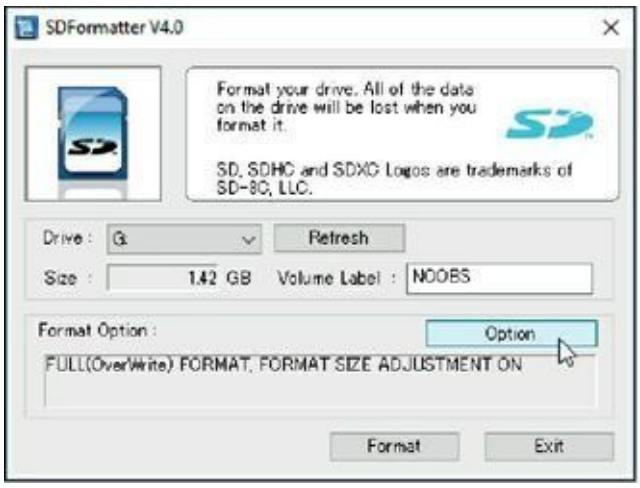


Figure 2.10 : La fenêtre principale de l'outil SDFormatter sous Windows.

Vérifie plutôt deux fois qu'une que tu as sélectionné la bonne lettre de lecteur correspondant à la carte. Aide-toi de la valeur indiquée dans la boîte de taille `Size`. Si elle indique une taille importante, c'est que tu es en train de te préparer à supprimer la totalité du contenu de ton disque système Windows ! Tu serais très déçu si cela arrivait. Autrement dit, une fois que tu as revérifié, revérifie encore une fois.

4. Dans la zone de nom de la carte `Volume label`, indique `NOOBS`.
5. Clique le bouton `Options`.
6. Dans la sous-boîte des options, ouvre la liste `FORMAT TYPE` et sélectionne `FULL (Overwrite)`.

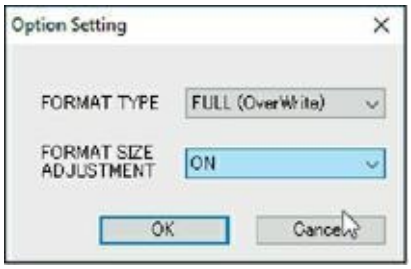


Figure 2.11 : La boîte des options de formatage Windows.

7. Toujours dans cette boîte d'options, ouvre la liste `FORMAT SIZE ADJUSTEMENT` et sélectionne `ON`.
8. Clique OK.
9. Revérifie une dernière fois que tu as choisi la bonne lettre qui correspond à la carte mémoire et non à un disque dur externe bourré de photos et autres documents personnels qui disparaîtraient à tout jamais.

10. Une fois que tu es certain d'avoir bien sélectionné ta carte mémoire, clique enfin le bouton `Format`.
11. Tu peux aller vaquer à d'autres occupations pendant le formatage.
12. Clique le bouton OK pour refermer la fenêtre.
13. Quitte enfin l'outil avec le bouton `Exit`.

Téléchargement de NOOBS sous Windows

Le fichier NOOBS est strictement le même sous Windows et sous macOS.

1. Rends-toi à l'adresse suivante pour télécharger le fichier compressé :

   `www.raspberry.org`

2. Dans la page d'accueil, clique en haut le bouton `DOWNLOADS` pour accéder à la page des téléchargements.

   Tu dois trouver une icône pour NOOBS.

3. Clique l'icône `NOOBS` de gauche (`Offline and network install`).

Je rappelle qu'il faut choisir la variante standard. Prends bien note de l'endroit ou tu fais déposer le fichier téléchargé.

Transfert du contenu sur la carte

Une fois le téléchargement terminé, il ne reste plus qu'à décompresser le contenu et à le copier.

1. Dans le Gestionnaire de fichiers, double-clique le nom du fichier téléchargé.

   Tu vois apparaître une page montrant le contenu de l'archive.

2. Ouvre une autre fenêtre de Gestionnaire de fichiers et affiche le contenu de la carte mémoire (le panneau droit doit être vide).

3. Reviens à la première fenêtre pour sélectionner tous les fichiers contenus dans l'archive. Tu peux alors copier par glisser-déposer tous ces fichiers depuis la première fenêtre vers celle de la carte mémoire (Figure 2.12).

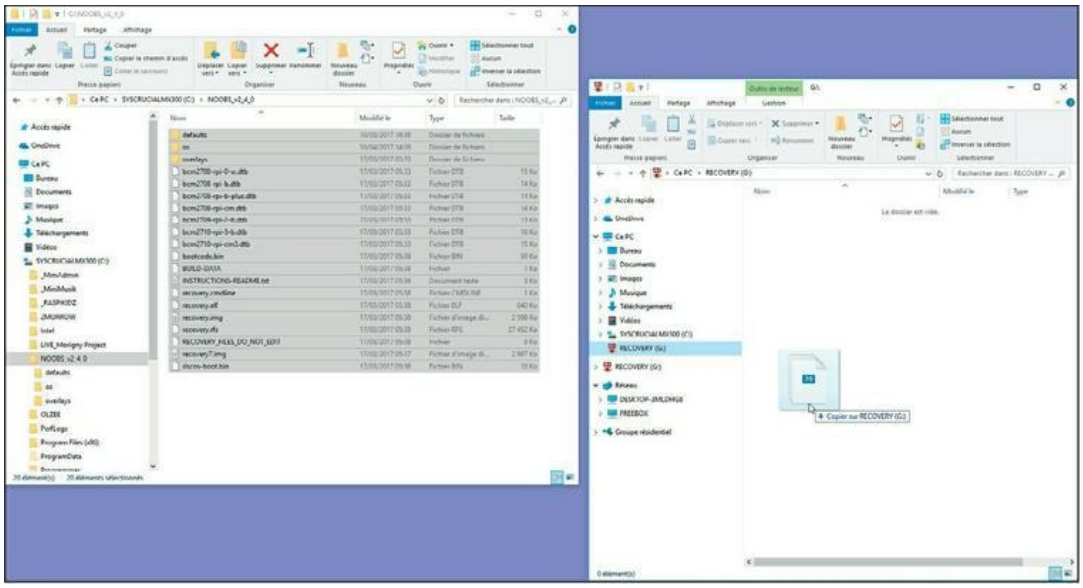


Figure 2.12 : Transfert des fichiers NOOBS vers la carte.

4. Une fois la copie terminée, procède à l'éjection de la carte de la manière dont tu as l'habitude de le faire.

En général, la technique la plus simple consiste à utiliser l'icône de périphérique dans la partie droite de la barre d'état en bas (Figure 2.13).

Figure 2.13 : Éjection de la carte mémoire.

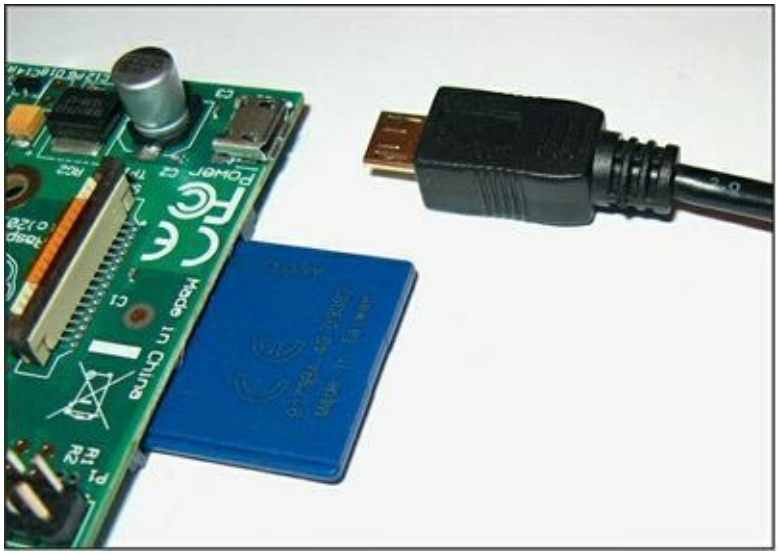
5. Vérifie que la carte mémoire n'apparaît plus dans la liste des périphériques du Gestionnaire de fichiers.

Une fois que la carte est éjectée, procède à son extraction physique. Elle est prête à servir à donner vie à ton Raspi !

Chapitre 3
Branche-moi !

Les branchements du Raspberry ne sont pas difficiles, mais ils peuvent être assez nombreux. Si tu prévois de ne plus déplacer ton nano-ordinateur, tu n'auras à réaliser ces branchements qu'une fois. Voici les branchements que tu peux prévoir, tous n'étant pas obligatoires :

- l'alimentation en énergie ;
- la sortie HDMI vers l'écran ;
- un branchement pour le clavier ;
- un branchement pour la souris ;
- un branchement vers le réseau Ethernet ;
- un branchement pour la sortie audio (option) ;
- l'insertion de la carte SD contenant le système.



Choisir un lieu de résidence

Si tous les ingrédients de ton système sont prêts, tu peux te lancer dans les branchements. Si ce n'est pas le cas, revois la liste fournie dans le Chapitre 1.

Commence par dégager un peu de place sur un bureau, afin de pouvoir y positionner l'écran, une zone libre pour la souris et pour le clavier. Prends de la marge, parce qu'il faut toujours plus de place que prévu au départ.

Si tu n'as aucun bureau ou table disponible, tu peux envisager de positionner ton Raspberry au sol, mais obligatoirement dans un boîtier approprié. Tu peux également envisager d'installer ton Raspi dans un boîtier accroché au dos d'un téléviseur grâce à un support à la norme VESA. Ce sera idéal si tu comptes utiliser le Raspberry comme centre multimédia.

Certains voudront découvrir leur nouveau nano-ordinateur sur la table de la cuisine. Il est vrai qu'en dehors des heures de repas et de leur confection, ce genre de plan de travail offre suffisamment d'espace. Si tu n'as vraiment pas d'autre endroit, mets-toi d'accord avec les autres utilisateurs de cet espace hautement stratégique pour savoir dans quelle tranche horaire tu vas pouvoir l'occuper.

Trouver une source d’énergie

Bien sûr, tu es toi-même une source d’énergie fantastique, et tu vas acquérir de superpouvoirs quand tu auras appris à programmer.

Plus prosaïquement, ton ordinateur a besoin d’électricité, et dans notre cas, tu vas devoir trouver plusieurs prises électriques.

Il te faut donc dans tous les cas une multiprise avec au moins 3 prises. Si la prise de courant la plus proche est trop loin, prévois également une rallonge.

Au niveau des besoins en courant, ils sont minimes. Le circuit Raspberry Pi a besoin au maximum de 2,5 A, mais sous 5 V, ce qui représente moins de 100 milliampères en 220 V. L’écran va consommer un peu plus, mais moins de 100 watts.

Commence donc par mettre en place la multiprise, par exemple sur le côté du bureau ou de la table. Cela t’évitera de devoir ramper à quatre pattes derrière le bureau pour brancher ou débrancher quelque chose (d’autant que tu pourrais bien te cogner la tête en te relevant).

On trouve à des prix raisonnables de nos jours des multiprises avec protection contre la foudre. Certaines sont même dotées d’une ou deux prises USB pour recharger des appareils. S’il y a une fonction de filtrage, cela peut s’avérer utile. En revanche, au niveau de la protection contre la foudre, si ton logement est frappé, il y a sans doute d’autres objets plus précieux à protéger d’abord.

Insérer la carte mémoire

Je conseille d’insérer la carte mémoire en premier, parce que tu seras moins gêné par les autres câbles.

Une particularité du Raspi est que le porte-carte est soudé sur l’autre face de la plaque. La carte mémoire va déborder un peu, même totalement insérée. Surtout, elle doit être insérée à l’envers. Autrement dit, les contacts doivent être du côté du circuit.

Du fait que la carte continue à déborder un peu, elle peut sembler fragile. Il faut en tenir compte.

Insertion de la carte dans un modèle A ou B

Les anciens Raspberry acceptaient une carte mémoire SD grand format. Commence par positionner la carte en sorte que les composants électroniques soient visibles sur le dessus. Bien sûr, le Raspi ne doit pas être sous tension.

Prends la carte mémoire et retourne-la pour voir les contacts. Insère la carte côté contacts dans le porte-carte qui se trouve sur le dessous de la plaque jusqu'à arriver en butée.

Si cela ne peut ou ne veut pas entrer, ne force surtout pas. Vérifie que les contacts sont bien visibles sur le dessus quand tu procèdes à l'insertion.

Retourne la plaque pour vérifier que la carte est bien en place. Il ne doit plus rester d'espace entre la partie insérée et le porte-carte (Figure 3.1).

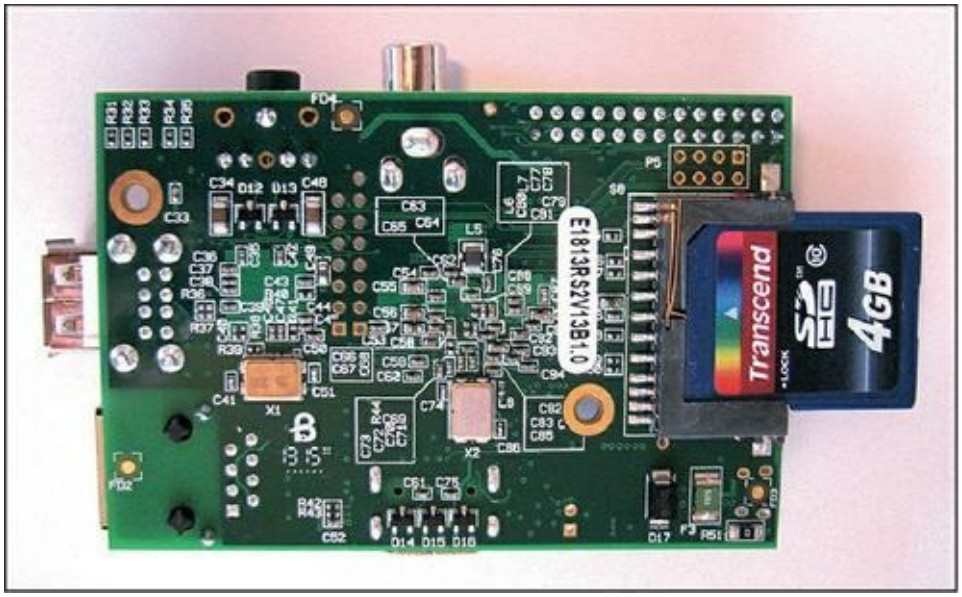


Figure 3.1 : La carte mémoire insérée dans un modèle B.

Tu peux maintenant retourner la plaque dans le bon sens.

Sur les anciens modèles A et B, la carte n'est pas vraiment retenue. Si tu la manipules un peu, elle peut facilement ressortir du logement, ce qui est une mauvaise chose.

Essaye de ne jamais déplacer le Raspi quand il est sous-tension et qu'un logiciel est en cours d'exécution. Ne touche pas la carte mémoire. Enfin, n'insère pas la carte après avoir mis le Raspi sous tension, et ne l'éjecte pas tant qu'il n'est pas éteint.

Une éjection sauvage risque d'endommager la carte mémoire et peut-être même le Raspberry. Cela te rendrait très triste et je ne veux pas que cela puisse se produire.

Insertion de la carte mémoire dans un modèle récent

Pour les générations de Raspberry A+, B+, et toutes les plus récentes, il y a une énorme différence : le porte-carte mémoire est au format micro USB. Je te conseille de retourner voir dans le Chapitre 1 à quoi ressemblent les différents formats de cartes mémoire.

L'avantage de ce nouveau porte-carte est que la carte reste verrouillée et ne risque pas de s'éjecter par accident.

Lorsque tu as besoin d'extraire la carte, tu commences par débrancher l'alimentation du circuit puis tu tires sur la carte pour la faire sortir. Il faut tirer avec deux doigts pour la récupérer.

Brancher un écran

Tu peux brancher un écran informatique ou un téléviseur à écran plat. Positionne l'écran sur le bureau puis retourne-le pour voir les connecteurs à l'arrière.

Si tu as lu le Chapitre 1, tu sais qu'il y a deux possibilités. Du côté du Raspi, la sortie pour affichage se fait sur un connecteur HDMI. Du

côté de l’écran, tu peux soit avoir un connecteur HDMI, soit un connecteur DVI.

Avec un câble HDMI vers HDMI

Si ton écran possède une entrée HDMI, ce que tu peux facilement vérifier parce qu’il y a la mention HDMI à côté du connecteur, il suffit d’utiliser un câble normal et de le brancher dans cette entrée et dans le connecteur HDMI du Raspi.

Dans la Figure 3.2, je montre l’extrémité d’un câble HDMI. L’autre extrémité est déjà en place sur le Raspi.

Figure 3.2 : Le câble HDMI.

Sur les premiers modèles de Raspberry, il y avait également une sortie vidéo analogique qui permettait de se brancher sur un écran cathodique. Ces écrans étant devenus extrêmement rares, la sortie vidéo composite n’est plus proposée dans les plus récents modèles.

Le connecteur pour écran est quasiment au même endroit sur les modèles plus récents à partir des A+ et B+.

Les gros boîtiers en métal brillant sur la plaque sont, je le rappelle, pour le connecteur réseau Ethernet et pour les appareils USB. Tu peux les voir dans la Figure 3.2 parce qu'ils sont situés à côté de la prise HDMI.

Branchement d'un écran DVI

Si ton écran possède une entrée numérique DVI, il faut te doter d'un adaptateur ou d'un câble HDMI vers DVI (j'en ai parlé dans le Chapitre 1).

Le côté du connecteur DVI est visible sur la Figure 3.3. Le côté HDMI se branche au Raspi et l'autre à l'écran.

Il y a une énorme différence entre les deux types de connecteurs : une fois inséré, le connecteur HDMI reste en place. En revanche, le connecteur DVI ne tient pas en place, sauf si tu peux visser les deux molettes de fixation.

Tu remarques les deux vis sur le connecteur DVI. Prends la peine de les utili ser pour que la prise ne se débranche pas au mauvais moment.

Figure 3.3 : Un câble DVI.

Allumage de l’écran

Il ne reste plus qu’à brancher le cordon secteur de l’écran à ta multiprise.

Tu peux ensuite mettre l’écran sous tension. Il n’affiche rien d’intéressant pour le moment.

(Option) Ajout d’un concentrateur USB (hub)

Si tu possèdes un ancien modèle A ou B, il te faut absolument un concentrateur USB, c’est-à-dire une sorte de multiprise USB.

Si tu possèdes un modèle plus récent, Pi 2 ou Pi 3, tu peux passer directement à la section suivante.

Les concentrateurs USB sont proposés dans une grande variété d’aspects. Certains sont tout simplement rectangulaires, d’autres constituent une petite tour, d’autres encore sont en forme d’animal. La grande différence est que certains sont statiques alors que d’autres sont autoalimentés. Il te faut absolument un USB qui dispose de sa propre alimentation.

Celui représenté dans la Figure 3.4 montre le connecteur dans lequel vient se brancher la prise de son adaptateur secteur. Pour tout détail, je t’invite à revenir au Chapitre 1.

Commence par brancher le câble USB de l’adaptateur dans une des prises USB de ton Raspi. Toutes ont le même usage.

Branche ensuite l’alimentation électrique du concentrateur. Pour certains modèles, le fil d’alimentation est directement soudé au boîtier.

Ton concentrateur est prêt à recevoir les périphériques USB que nous allons relier à notre Raspberry. Du fait qu’il est alimenté, les périphériques ne vont pas soutirer de l’énergie au Raspi.

Figure 3.4 : Branchements d'un concentrateur USB.

En général, la tension de sortie de l'adaptateur électrique est indiquée sur l'étiquette, par exemple 5 V DC, ce qui signifie qu'il fournit en sortie une tension de 5 V en courant continu (*Direct Current*). Je te conseille d'ajouter une étiquette collante ou un bracelet autour du fil pour ne pas mélanger cet adaptateur avec les nombreux autres que tu as chez toi.

Si tu utilises un ancien modèle A ou B, tu dois brancher tous les périphériques USB au concentrateur, et seul le concentrateur doit être branché au Raspberry pour ne pas lui voler trop de courant, ce qui empêcherait le circuit de fonctionner.

Brancher un clavier et une souris

Ces deux branchements sont faciles. Si tu disposes d'un modèle récent, A+, B+, Pi 2 ou Pi 3, tu branches le périphérique à une prise USB du Raspberry.

Dans le cas d'un modèle ancien, tu dois brancher les deux périphériques à ton concentrateur USB.

Une fois que tu as réalisé tes branchements, tu vas constater que la surface de travail commence à être encombrée de fils. Je te conseille d'organiser un peu tout cela en réunissant les câbles avec des colliers ou des petits bouts de ficelle.

Brancher la connexion réseau Ethernet

De nos jours, il y a un routeur Ethernet dans la plupart des domiciles, combiné à la fameuse « box » . Ton routeur doit sans doute offrir au moins une prise USB disponible.

Pour relier ton Raspberry à Internet, il suffit donc de tirer un câble Ethernet entre le routeur et ton circuit. La prise réseau est la grosse prise à côté des prises USB.

Le branchement du côté du routeur doit ressembler à ce que montre la Figure 3.5. Bien sûr, chaque routeur est différent, mais cette différence n'a aucune importance. L'essentiel est que le câble ne soit pas un câble croisé.

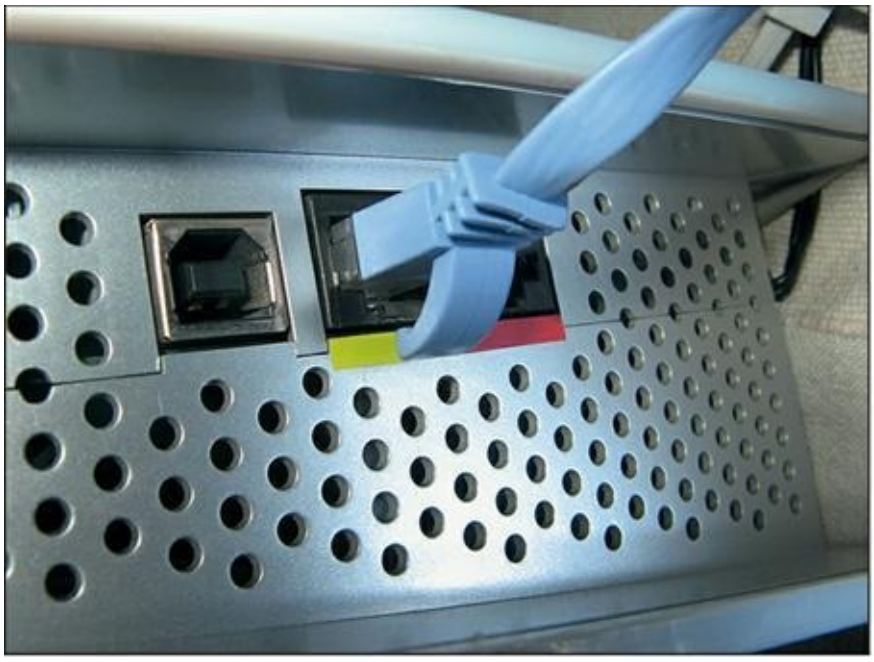


Figure 3.5 : Branchement du câble Ethernet au routeur.

Brancher l’adaptateur secteur

La Figure 3.6 montre à quoi ressemble un adaptateur secteur pour Raspberry Pi. De nos jours, les adaptateurs sont livrés avec plusieurs plaques de fiches pour s’adapter aux prises secteur au format européen, britannique, américain et japonais. Choisis la contrefiche appropriée à tes prises.

Certains adaptateurs offrent une ou deux prises USB, et d’autres encore sont solidaires du câble USB d’alimentation.

Dans tous les cas, le fonctionnement est identique. Il suffit d’insérer l’adaptateur dans la prise secteur.

Figure 3.6 : Un adaptateur secteur pour Raspberry.

Si ton adaptateur offre une prise USB supplémentaire, ne t’en sers pas. Il vaut mieux réserver toute l’énergie disponible au Raspi.

La Figure 3.7 montre où brancher l’autre extrémité du câble d’alimentation à ta carte Raspberry. Tu constates que le connecteur

est de type micro USB, généralement le même que celui des chargeurs pour téléphones.

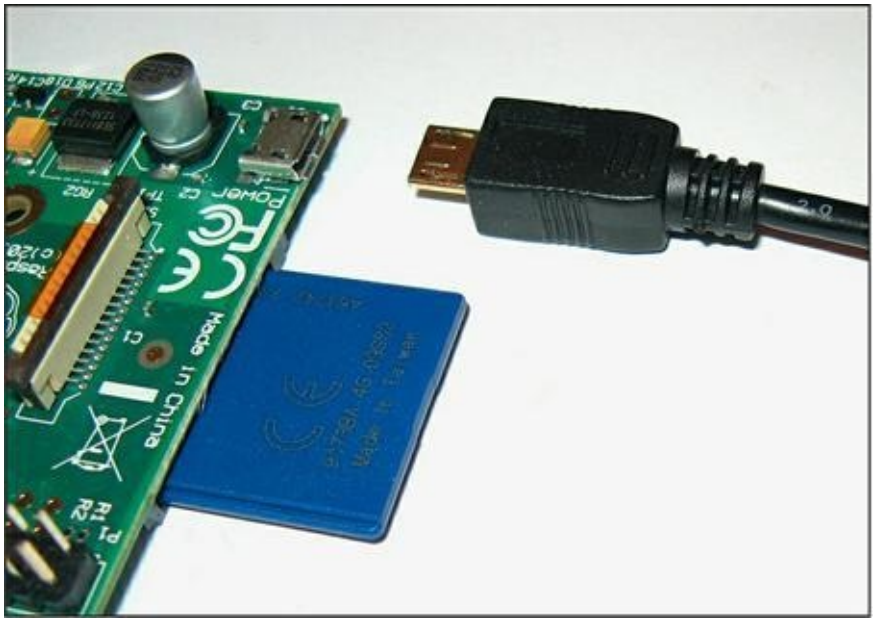


Figure 3.7 : Insertion du câble micro USB d'alimentation.

Tu as sans doute remarqué qu'il existe plusieurs formats de minifiches USB ; celle qui s'appelle miniUSB ressemblant beaucoup à la micro USB. Si tu comptes réutiliser un autre adaptateur, à condition qu'il offre une puissance suffisante, vérifie qu'il s'agit d'un connecteur micro USB. Ne force surtout pas pour insérer le connecteur dans le Raspberry. Il faut absolument un connecteur micro USB.

Sur les anciens modèles A et B, la prise d'alimentation était à côté du porte-carte mémoire. Sur les modèles plus récents, cette prise se trouve à côté du connecteur HDMI sur le haut de la plaque.

Si tu observes de près ton Raspberry, tu dois pouvoir lire la mention PWR IN si c'est un modèle récent, ou PWR pour un plus ancien. C'est écrit en très petites lettres.

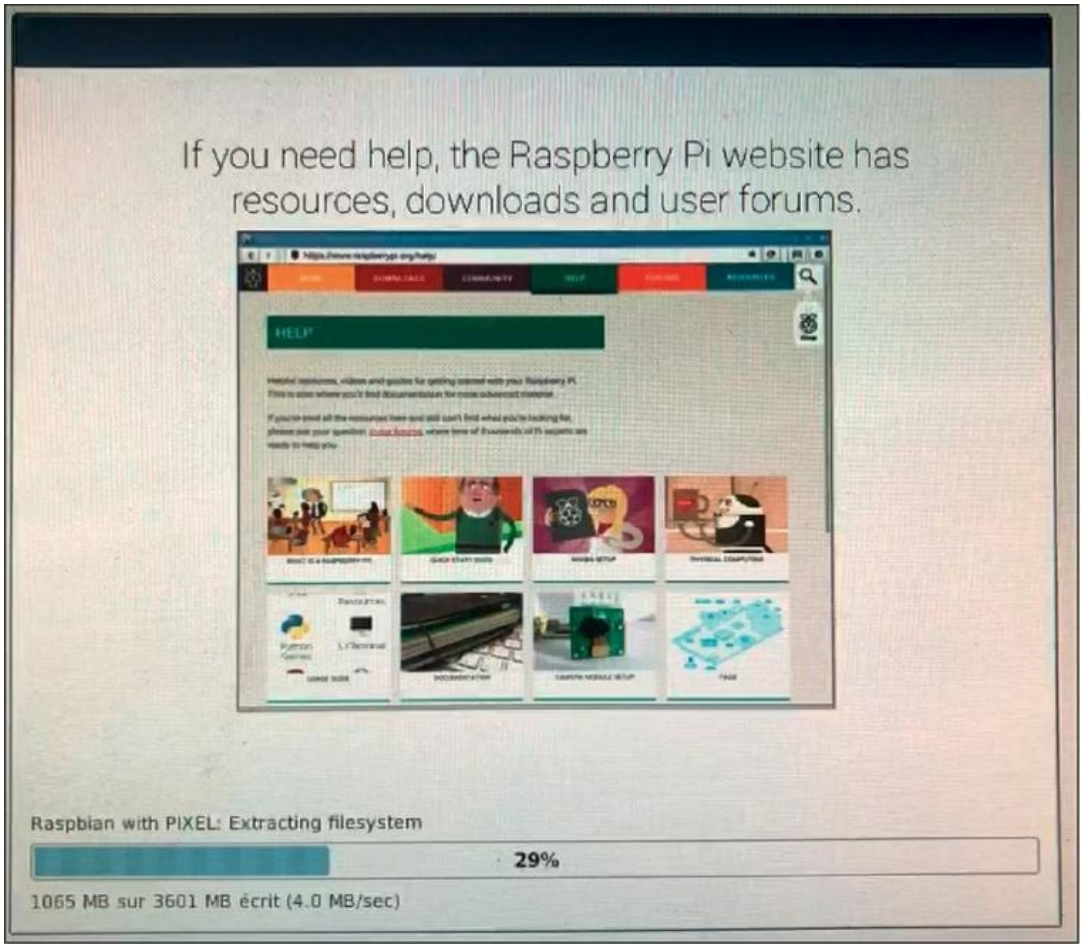
Ne cherche pas d'interrupteur d'alimentation. Les concepteurs du Raspberry ont décidé de ne pas en prévoir pour réduire les coûts et diminuer les risques de panne, et pour gagner un peu de place sur la plaque.

Te voilà prêt à démarrer pour la première fois ton Raspberry. Mais attends encore un peu. Il nous faut en savoir un peu plus au sujet de la configuration et de la procédure d'arrêt.

La prise micro USB du Raspberry n'est pas trop solide, par nature. Elle finira par lâcher si tu éteins ton Raspberry en le débranchant de façon répétée. Mieux vaut brancher et débrancher l'adaptateur du côté secteur. Mieux encore, utilise une multiprise secteur avec interrupteur. Une solution intermédiaire consiste à trouver une rallonge USB, ce qui permet de déconnecter l'alimentation au niveau de la rallonge, afin de ne jamais manipuler le connecteur micro USB sur la carte.

Chapitre 4
Démarre-moi !

Voir son nano-ordinateur s'éveiller pour la première fois est très excitant. Maintenant qu'il est branché, il ne te reste que quelques formalités pour commencer à l'apprivoiser. Rien de difficile en tout cas. Suis-moi.



Installation avec NOOBS

Le tout premier démarrage est différent des suivants : pour le moment, la carte mémoire n'est pas encore reconfigurée pour servir de disque de démarrage système. Ce qui va démarrer, c'est l'installateur/préparateur NOOBS. Ce n'est que quand il aura fini son travail que le système Raspbian sera opérationnel.

1. Une fois que tu as mis en place la carte mémoire contenant NOOBS, branche l'alimentation secteur du Raspi.

   Tu vois apparaître au bout de quelques instants la boîte de dialogue de la Figure 4.1.

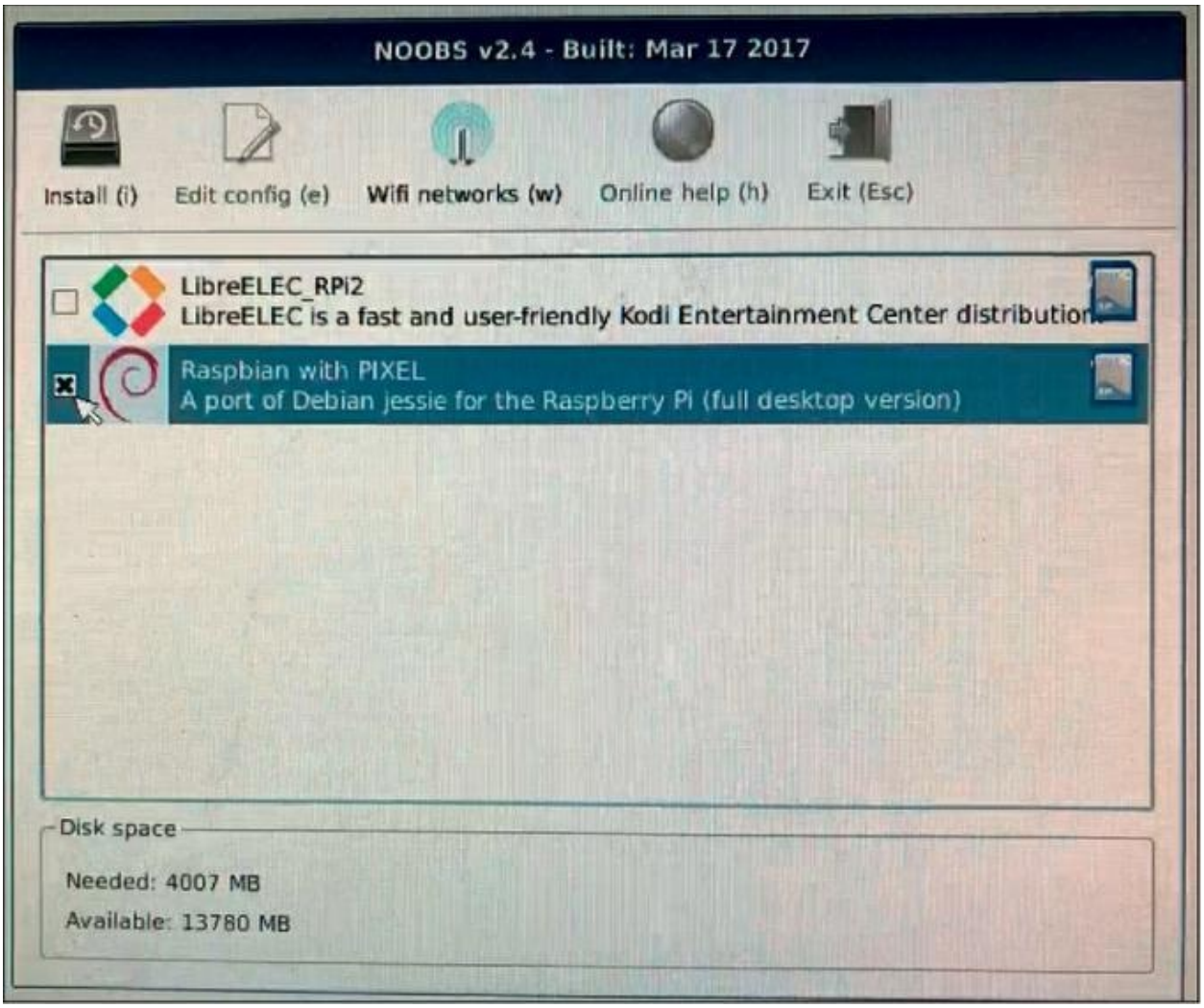


Figure 4.1 : Choix du système à installer.

Tu dois choisir le système que tu veux installer parmi au moins deux proposés. Dans notre cas, nous allons utiliser le système Raspbian, qui est la version de Debian spécialement personnalisée pour le Raspberry Pi.

2. Sélectionne donc l'option `Raspbian with PIXEL` en cochant la case à gauche.

Le bas de la boîte montre combien d'espace est nécessaire sur la carte, et combien d'espace est disponible.

3. Une fois que tu as vérifié que la carte offre assez d'espace, tu peux cliquer en haut à gauche le bouton `Install`.

4. Une boîte message apparaît pour te demander de confirmer. Lorsque la carte contient déjà un système, ce qui n'est pas le cas ici, cela t'évite d'écraser son contenu actuel par mégarde. Tu peux cliquer le bouton **Yes**.

   À partir de ce moment, toute la procédure d'installation automatisée va se dérouler sans que tu aies besoin d'intervenir (Figure 4.2).

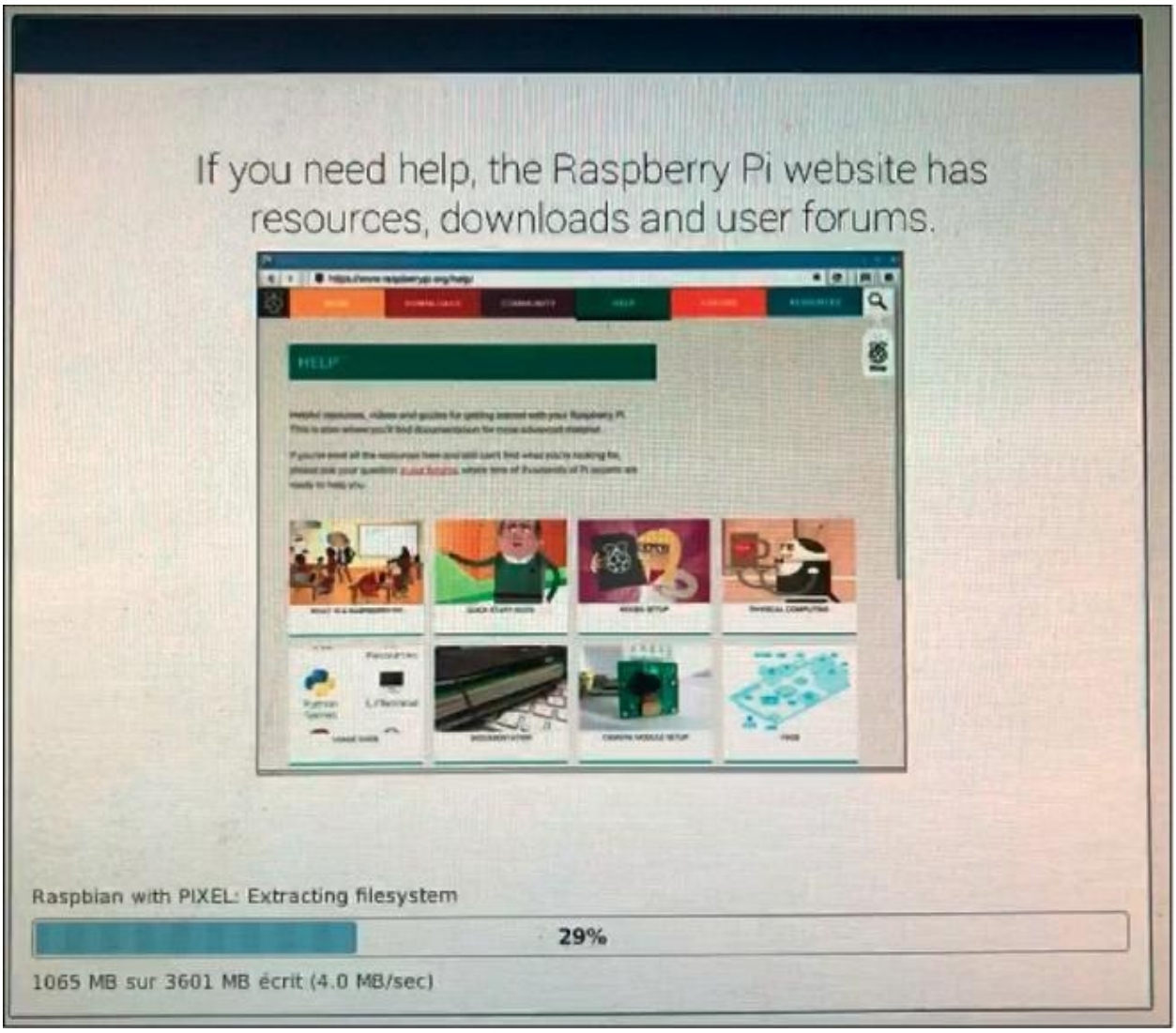


Figure 4.2 : Jauge d'avancement de l'installation.

En fait, tu peux en profiter pour préparer le système à fonctionner en français. En bas de l'écran, tu remarques en effet deux listes de sélection. Au départ, elles indiquent pour `Language` `English UK` et pour `Keyboard` `GB`.

5. Ouvre la liste `Language` et sélectionne `Français` (Figure 4.3).

Figure 4.3 : Sélection de la langue française.

6. Ouvre maintenant la liste de choix du clavier, **Keyboard**, et sélectionne, si ce n'est pas déjà fait, les deux lettres FR (pour France).

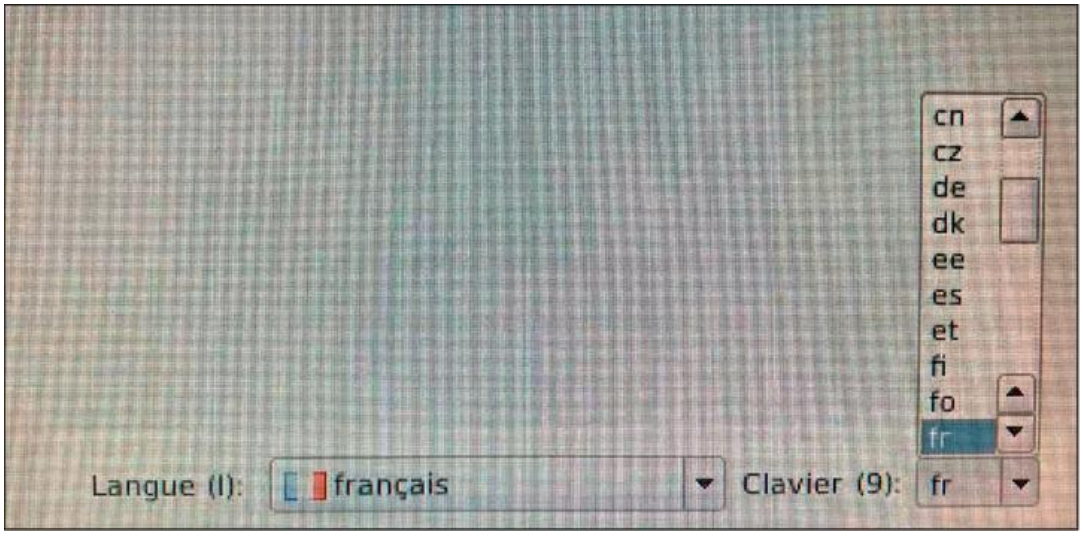


Figure 4.4 : Sélection du clavier français.

7. Tu peux maintenant laisser l'installation de Raspbian aller à son terme, ce qui peut prendre

entre 10 minutes et une demi-heure.

Une fois l'installation terminée, une boîte de message te le confirme. Clique Ok.

Figure 4.5 : Boîte de fin d'installation.

Le redémarrage est automatique. Tu vas dorénavant accéder à ton système d'exploitation Raspbian.

Premier démarrage de Raspbian

Quand le Raspi redémarre, tu vois d'abord apparaître une longue série de messages en mode texte sur fond noir puis le bureau graphique. Il ne comporte qu'une seule icône, celle de la poubelle, qui a son nom anglais pour l'instant, `Trash` (Figure 4.6).

Figure 4.6 : Bureau de Raspbian en anglais.

Nous allons donc immédiatement lui faire apprendre le français.

1. Clique l'icône dans le coin supérieur gauche, qui montre une framboise. C'est le menu général. Ouvre le sous-menu `Preferences` (à l'anglaise, sans accents) et sélectionne l'option `Raspberry Pi configuration` (Figure 4.7).

Figure 4.7 : Commande pour accéder à la configuration.

2. Tu vois apparaître la boîte de configuration qui comporte quatre pages. Au départ, tu es face à la première page, `System` (Figure 4.8).

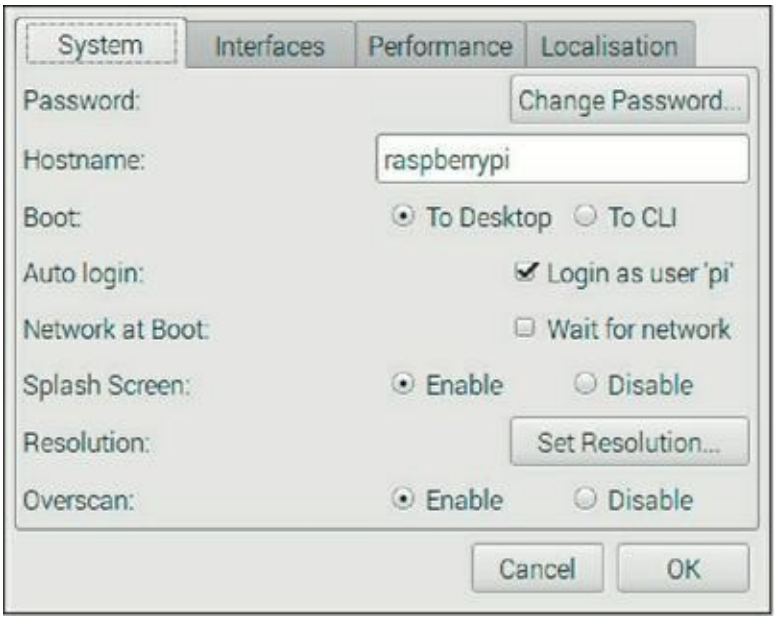


Figure 4.8 : Page de configuration System.

Nous n'allons rien changer dans cette page pour l'instant. Je te demande seulement d'observer l'option `Boot` qui est normalement réglée sur `To desktop`. Cela signifie que dès le démarrage, tu accèdes au bureau graphique.

Le choix de l'option `To CLI` fait démarrer le système en mode texte (CLI signifie *Command-Line Interface*). Dans ce cas, pour démarrer le bureau graphique, il faudra saisir la commande `startx`. Si l'identifiant et le mot de passe sont demandés, les voici :

Identifiant : `pi`

Mot de passe : `raspberry`

3. Clique l'onglet de la dernière page, `Localisation`. C'est ici que nous allons franciser notre nano-

ordinateur (Figure 4.9).

Figure 4.9 : Page de configuration Localisation.

4. Clique le premier bouton, `Set locale` (Figure 4.10).

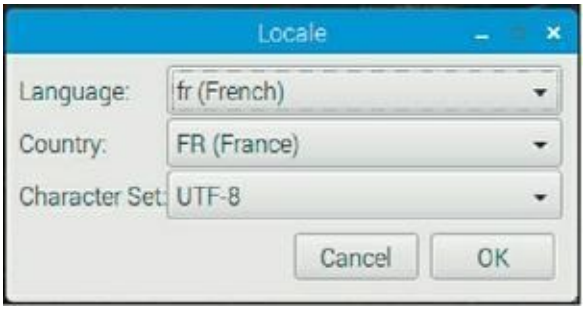


Figure 4.10 : Options de paramètres locaux.

Si tu as effectué le réglage proposé pendant l'installation du système en début de chapitre, tu constates que les paramètres sont déjà réglés pour la France. Si ce n'est pas le cas, sélectionne les trois options comme tu peux le voir dans la figure précédente et valide par OK.

5. Nous pouvons maintenant régler le fuseau horaire au moyen de la deuxième option. Clique le bouton `Set Timezone` (Figure 4.11).

Figure 4.11 : Réglage du fuseau horaire.

La plupart des lecteurs pourront choisir `Europe` pour la liste `Area` et sans doute `Paris` pour la position `Location`. Les lecteurs francophones qui sont situés ailleurs sur notre bonne vieille Terre choisiront par exemple le `Canada` et `Montréal`. Referme la boîte par OK.

Nous pouvons maintenant choisir la configuration du clavier afin de ne plus saisir des lettres différentes des touches clavier (notamment le A, le Z, le W, *etc.*). Clique donc le bouton `Set Keyboard` (Figure 4.12).

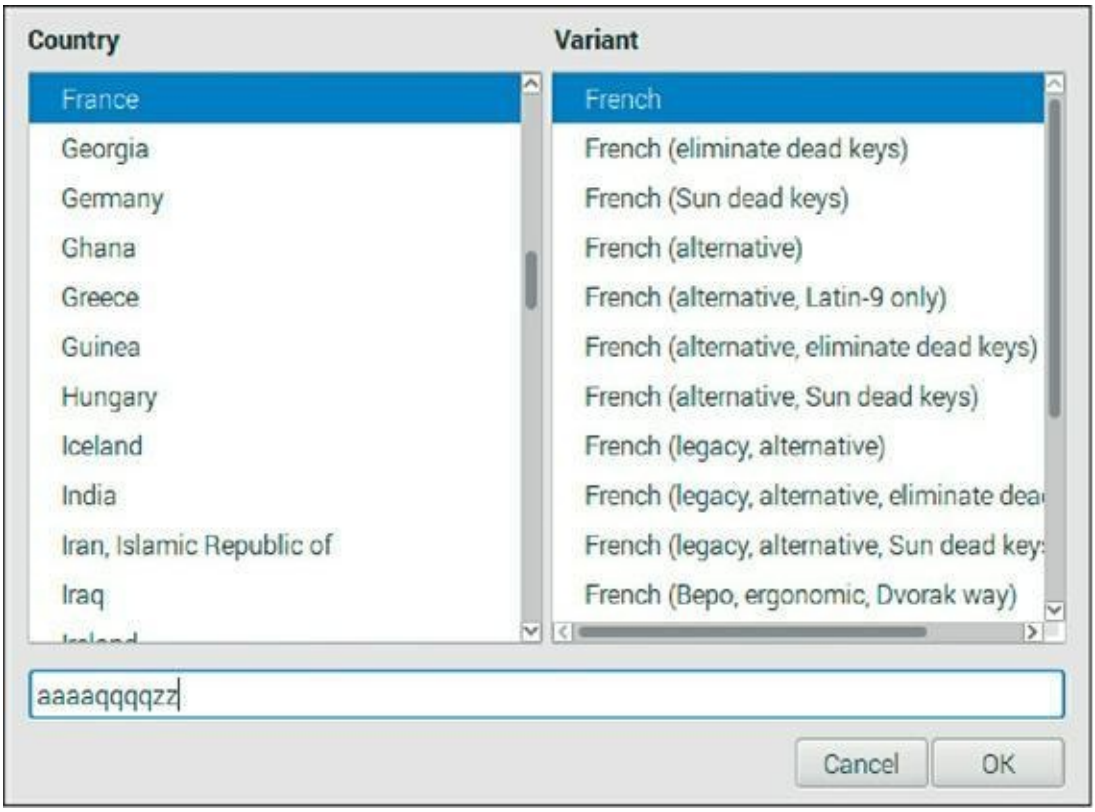


Figure 4.12 : Réglage de la configuration du clavier.

Tu constates peut-être que le clavier est réglé sur `United Kingdom` alors que nous avions demandé un clavier français pendant l'installation. Cet oubli sera sans doute corrigé à l'heure où tu vas lire ces lignes.

6. Dans la liste de gauche, utilise l'ascenseur pour remonter jusqu'à `France`. Dans la liste de la variante à droite, tu peux normalement te contenter de la première option présélectionnée. Clique dans la zone d'essai en bas et teste notamment les touches des lettres A, Z, les chiffres au-dessus du clavier et les signes de ponctuation. Referme ensuite la boîte par OK.

7. Il ne reste plus qu'à régler éventuellement le code pays si tu comptes utiliser le Wifi. Clique le bouton `Set Wifi Country` puis sélectionne ton pays, sans doute FR pour la plupart des lecteurs (Figure 4.13).

Figure 4.13 : Sélection du pays WiFi.

8. Il ne reste plus qu'à refermer par OK la boîte principale de configuration.

   Tu vois alors apparaître une boîte avec un message en anglais qui te rappelle que tes choix ne seront appliqués qu'après le prochain redémarrage. Réponds par `Yes` puisqu'il faut redémarrer.

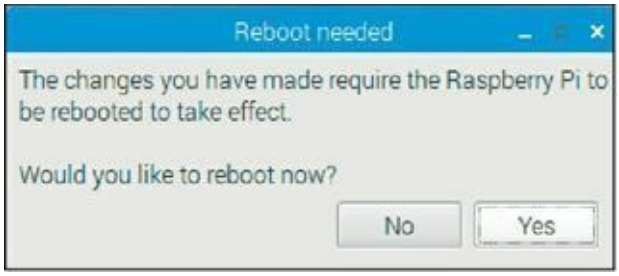


Figure 4.14 : Demande de redémarrage.

Configuration en mode texte

Toute la procédure de configuration que nous venons de réaliser utilise des fenêtres, des boîtes de dialogue et des boutons, bref, une interface moderne. Toutes ces options peuvent tout aussi bien être

modifiées depuis la ligne de commande, donc en mode texte. Tu n'en as pas besoin pour l'instant, mais il est utile de savoir comment accéder à cet outil.

Lorsque tu seras dans le mode texte, soit parce que tu as demandé à démarrer dans le bureau, soit dans une fenêtre de terminal, il suffit de saisir cette commande et de valider par Entrée :

```
sudo raspi-config
```

Tout doit être saisi en minuscules et le nom de l'outil comporte un tiret. Tu vois alors apparaître un menu en mode texte qui propose les catégories d'options vues plus haut, et quelques autres plus techniques.

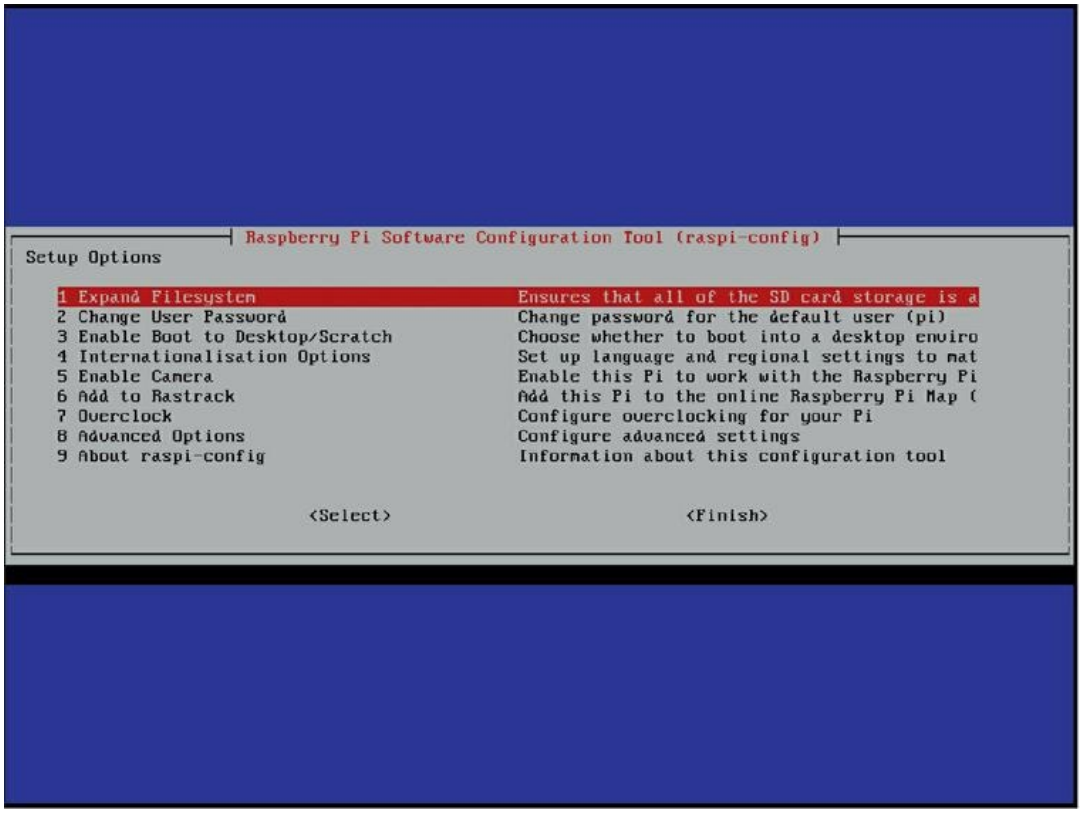


Figure 4.15 : L'outil de configuration en mode texte.

Normalement, tu n'auras pas besoin d'utiliser cet outil.

Arrêter proprement et redémarrer

Dans un ordinateur, aussi petit soit-il, il se passe toujours quelque chose, même quand toi, en tant qu'utilisateur, tu n'interagis pas avec, ni au clavier, ni à la souris.

Autrement dit, il est absolument interdit de couper le courant pour arrêter la machine, sauf quand il n'y a plus rien d'autre à faire pour ramener la machine à la raison.

En effet, le système Linux travaille en coulisses en permanence, notamment en réorganisant les données sur la carte, en démarrant des processus, et en en arrêtant d'autres. Cette activité est en général invisible, mais le fait de l'interrompre brutalement oblige le système à procéder à un nettoyage lors du prochain redémarrage. Pendant ce temps, tu dois patienter.

En demandant au système de s'arrêter, tu lui laisses le temps de réaliser ses opérations de clôture et tu t'épargnes la phase de nettoyage au redémarrage.

Arrêt depuis le bureau en mode graphique

Dans l'interface graphique, il suffit d'ouvrir le sous-menu `Shutdown` du menu général puis de choisir entre arrêt complet (`Shutdown`) et redémarrage (`Reboot`).

La fiche micro USB de l'alimentation ne va pas supporter des centaines de cycles d'extraction/insertion. Pour éviter tout souci, n'éteins pas ton Raspi en débranchant cette fiche. Retire plutôt la fiche secteur de l'alimentation ou utilise une multiprise secteur avec interrupteur.

Avant de couper l'alimentation, il faut attendre encore quelques instants que le témoin lumineux sur la carte ne clignote plus.

Le témoin rouge sur la carte reste allumé tant qu'il y a de la tension en provenance de l'alimentation. Lorsque c'est le seul allumé, tu peux éteindre.

Arrêt en mode texte

En mode texte, le redémarrage correspond à cette commande :

```
sudo reboot
```

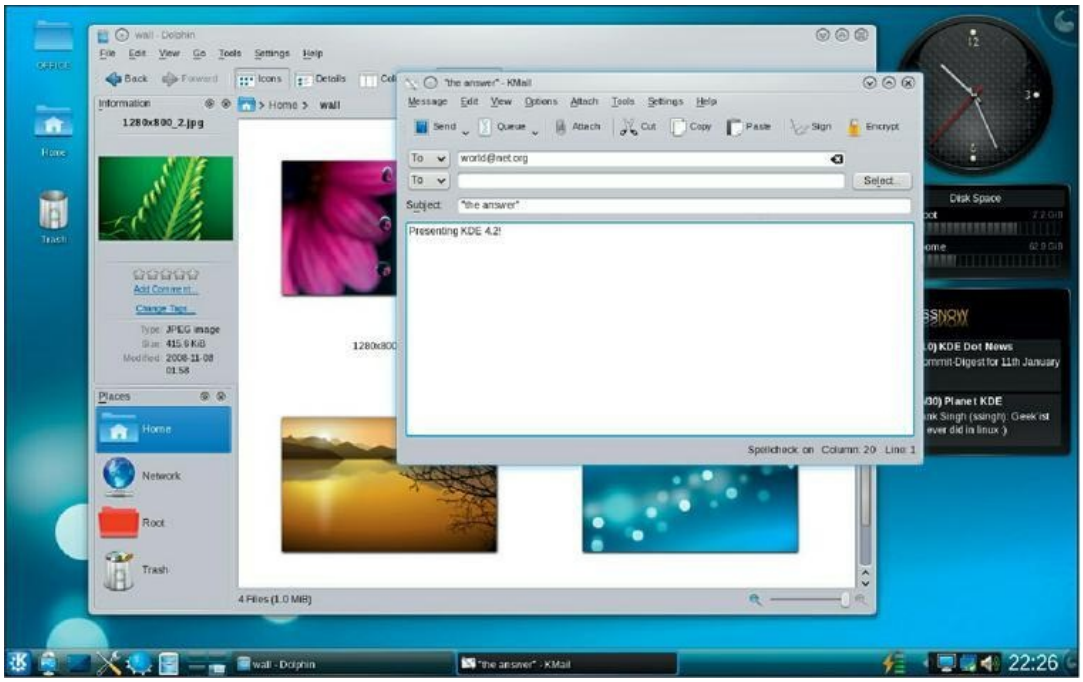
L'arrêt complet en mode texte se demande avec cette commande :

```
sudo shutdown -h now
```

L'option `-h` signifie *halt*.

Chapitre 5
Le bureau graphique

Ce que j'appelle ici le bureau graphique correspond sous Windows comme sous macOS au Bureau. J'ajoute l'adjectif « graphique » volontairement, car il est assez fréquent sur le Raspi de rebasculer dans le mode texte des anciens ordinateurs pour travailler plus efficacement. Le bureau graphique du Raspi fonctionne sur les mêmes principes que celui que tu connais déjà sous Windows ou sous macOS. Tu n'auras pas à bousculer tes habitudes d'utilisateur.



Découvrir le bureau

Si le Raspi se trouve en mode texte après avoir démarré (c'est peu probable dans les plus récentes versions de Raspbian), tu peux démarrer le bureau graphique au moyen de la commande `startx`

au niveau de l'invite de commande. Je rappelle que pour ouvrir une session sur ta machine (t'identifier), tu saisis le nom d'utilisateur prédéfini `pi` et le mot de passe `raspberry` (tout en minuscules).

Une fois le bureau graphique chargé (c'est une application presque comme une autre), tu vois apparaître l'écran montré en Figure 5.1. À partir de ce moment, tu peux utiliser comme bon te semble ta souris pour déplacer, ouvrir et refermer des fenêtres, lancer des applications et faire tout ce que tu as l'habitude de faire sur un bureau.

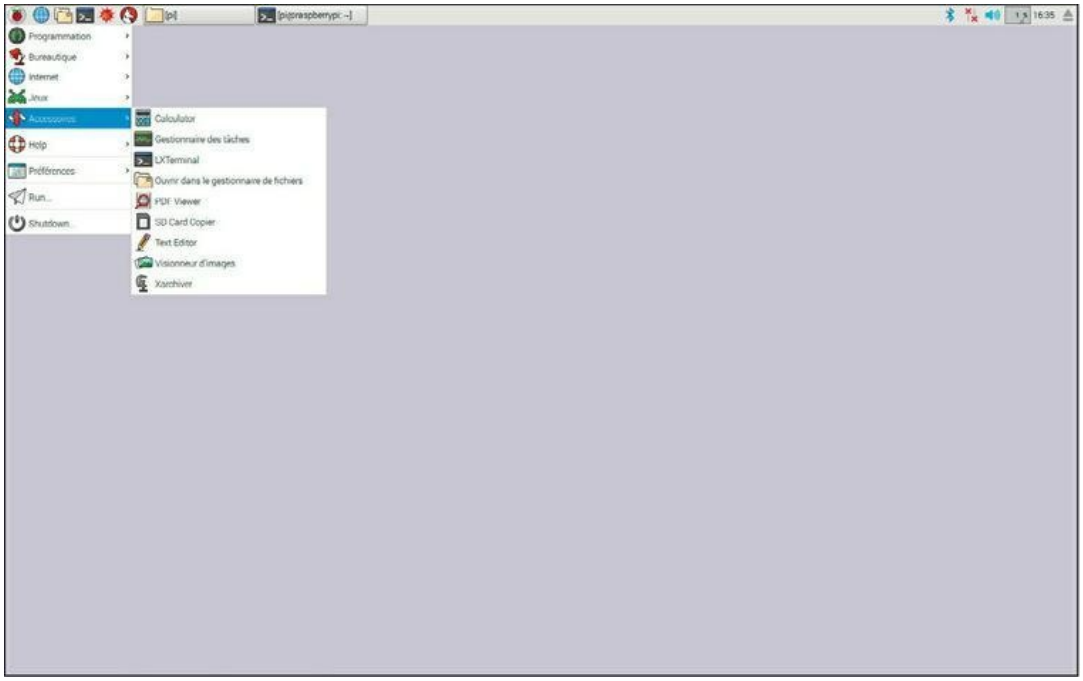


Figure 5.1 : Aspect général du bureau graphique de Raspbian.

Le bureau du Raspi est une application d'interface utilisateur qui porte le nom LXDE (*Lightweight X 11 Desktop Environment*). Tu peux la remplacer par une autre application qui a le même rôle. Les deux autres interfaces utilisateur très répandues sont Gnome et KDE (Figure 5.2). Toutes deux offrent bien plus de possibilités que LXDE, mais elles sont plus gourmandes en puissance. C'est la raison pour laquelle les concepteurs du Raspberry Pi ont préféré installer d'office l'interface LXDE, qui conviendra à presque à tous tes besoins.

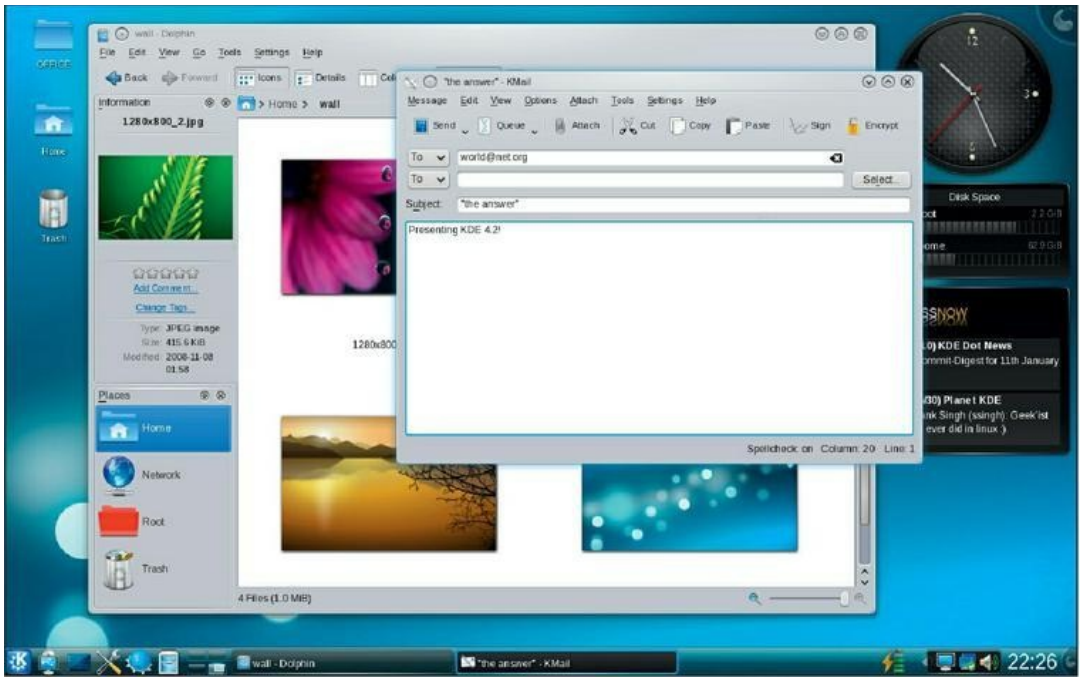


Figure 5.2 : Aspect général du bureau graphique de KDE.

Les anciennes versions du système du Raspi étaient livrées avec une ancienne version de LXDE qui ressemblait plus à un bureau Windows, avec un certain nombre d'icônes déjà présentes sur le bureau et le bouton du menu général en bas à gauche

Une visite guidée du bureau

Que voit-on dès le départ sur ce bureau ? Tu peux remarquer deux zones principales :

- la grande surface du bureau ;
- la barre des tâches en bas.

Position de la barre des tâches

La barre des tâches est proposée au départ tout en haut de l'écran. Elle réunit du côté droit l'affichage de l'heure, quelques icônes d'application, notamment pour l'accès au réseau et à l'activité du

processeur. Le bouton du menu général est tout à gauche, symbolisé par une framboise.

La surface utile du bureau ne comporte au départ qu'une seule icône, celle de la Corbeille.

Bien gérer les fenêtres

Quand tu ouvres une fenêtre, elle vient se surimposer à ce qui est actuellement affiché. Tu peux la déplacer, la retailler et la minimiser.

Pour déplacer la fenêtre, agrippe-la avec le pointeur de souris dans la barre colorée du bord supérieur. C'est la barre de titre, car elle rappelle le nom de l'application qui possède cette fenêtre.

Pour retailler la fenêtre, tu amènes le pointeur de souris sur le bord désiré. Le coin inférieur droit de la fenêtre propose une retaille dans les deux dimensions à la fois.

Tu trouves enfin trois petits boutons en haut à droite de la fenêtre. Ce sont les boutons pour minimiser la fenêtre, la maximiser et la refermer.

Dans certains cas, le fait de fermer la fenêtre provoque la fin d'exécution du programme concerné. Lorsqu'une application possède plusieurs fenêtres, ce qui est le cas par exemple de Python, tu utiliseras de préférence la commande **Fichier/Quitter**.

Dès que tu ouvres une fenêtre, un onglet apparaît dans la barre des tâches. C'est cet onglet qui permet de réafficher la fenêtre lorsque tu la minimises.

La Figure 5.3 montre trois fenêtres d'application dans la barre des tâches, mais une seule est dépliée.

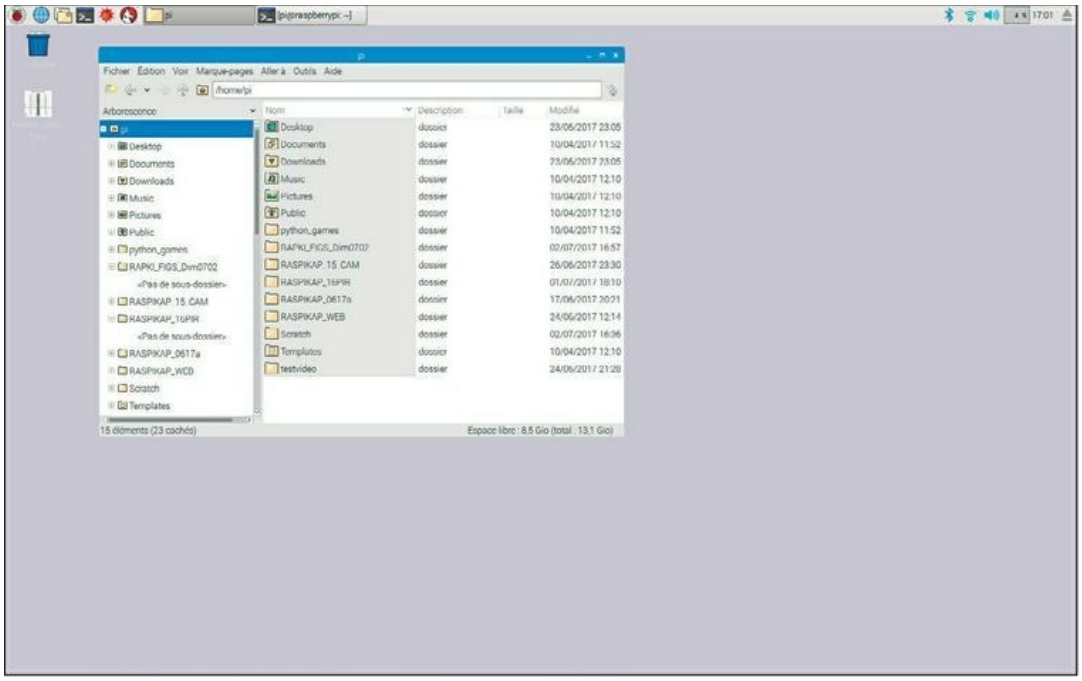


Figure 5.3 : Une fenêtre agrandie et deux fenêtres minimisées dans la barre des tâches.

Le choix du terme « bureau » pour décrire cette interface utilisateur découle de sa similarité avec un vrai bureau : il y a une corbeille, un affichage de l'heure, des dossiers que l'on peut ouvrir et refermer. Évidemment, sur un vrai bureau, tu ne peux pas modifier les dimensions d'une feuille de papier, ni exécuter un programme sur du papier. Mais la similarité reste suffisante pour que l'on ait choisi le terme bureau.

Le menu général

Tout en haut à gauche, tu vois l'icône d'une framboise qui correspond au menu principal. Il suffit de cliquer et de naviguer parmi les sous-menus pour choisir le programme dont tu as besoin. Si tu as déjà utilisé Windows ou macOS, tu devrais retrouver certains des éléments déjà connus.

La barre des tâches comporte une zone avec des icônes de lancement rapide. Elles sont très utiles lorsque l'icône normale qui a été créée

dans la surface du bureau est masquée par une fenêtre. Les icônes de lancement rapide sont toujours visibles.

Voici les icônes de lancement rapide présentes dès le départ :

- le navigateur Chromium ;
- le Gestionnaire de fichiers ;
- le terminal en mode texte LXterminal ;
- l'application Mathematica ;
- l'application Wolfram.

Les deux derniers programmes sont destinés aux étudiants et aux scientifiques, et je n'en parlerai pas plus dans ce livre.

En revanche, il est essentiel de savoir utiliser les trois premières icônes de lancement rapide pour bien profiter du bureau du Raspi.

Le navigateur Internet

La Figure 5.4 montre l'aspect général du navigateur Internet qui est installé dès le départ. Tu saisis l'adresse d'un site dans la zone d'adresse et tu disposes de plusieurs onglets. La figure montre la page d'accueil d'un nouveau moteur de recherche, qwant.com.

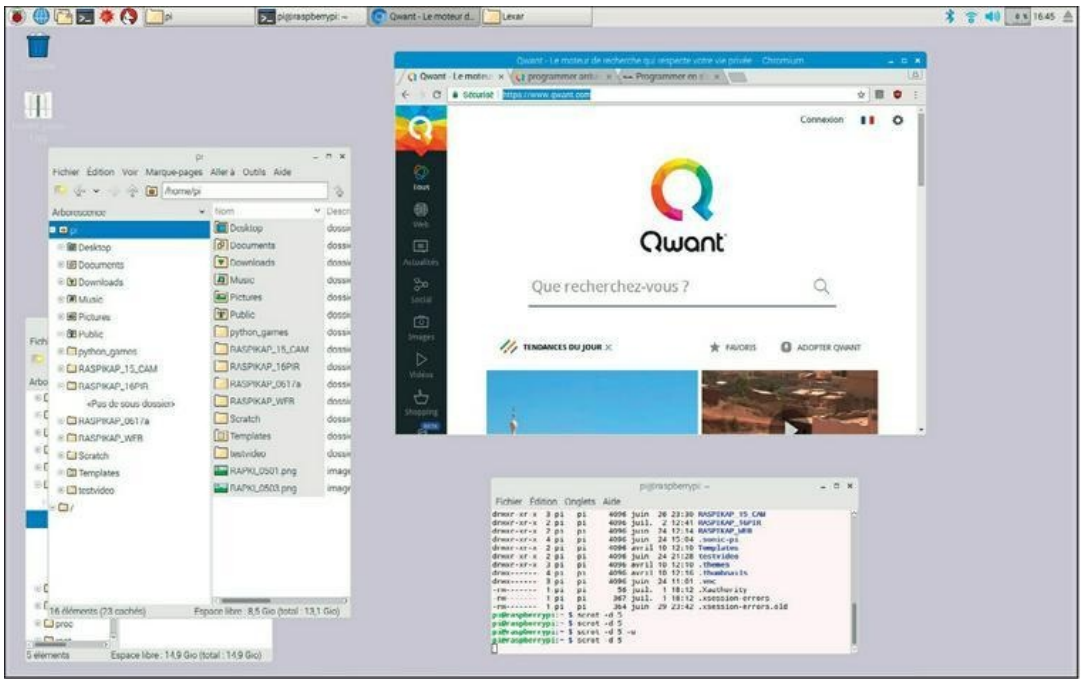


Figure 5.4 : Vue générale du navigateur Internet du Raspberry Pi.

Dans les précédentes versions du système, le navigateur proposé portait le nom Epiphany. Il était assez lent et a été heureusement remplacé par celui dont tu disposes maintenant.

Un autre navigateur ?

Il y a un second navigateur qui est installé de départ. Il s'appelle NetSurf. Il est un peu plus rapide que l'ancien Epiphany, mais il n'est pas très bon pour effectuer le rendu des pages Web riches. Certains mots vont notamment entrer en conflit avec d'autres à l'écran.

Si tu as envie de l'essayer, il faut d'abord savoir où il se cache. J'explique un peu plus loin dans ce chapitre dans quel sous-répertoire se trouve cette application.

Tu ne peux pas installer la plus récente version de n'importe quel navigateur tel que Chrome ou Internet Explorer. D'ailleurs, Chromium est une version spéciale de Chrome prévue pour le Raspi.

N. d. T. : Pour ne pas te rendre trop dépendant du moteur de recherche le plus utilisé de nos jours, Google, n'hésite pas à essayer le nouveau moteur de recherche Qwant qui a été conçu par des Français. Voici son adresse :

https://www.qwant.com/

Tu peux également utiliser le moteur Duck Duck Go (https://duckduckgo.com/).

Le Gestionnaire de fichiers

Les fichiers sont tous les conteneurs de données qui sont stockés sur la carte mémoire. Pour garantir une bonne organisation et une navigation facile, les fichiers sont répartis dans des répertoires et sous-répertoires (« répertoire » est le nom que l'on utilise sous Linux pour « dossier »).

Pour s'y retrouver dans cette énorme masse de données, tu disposes dans ton Raspi d'une application pour gérer les fichiers. Elle porte évidemment le nom *Gestionnaire de fichiers*.

Son icône est disponible dans la barre des icônes de lancement rapide. Elle ressemble un dossier papier. Clique cette icône pour lancer l'application et voir quelque chose dans le style de la Figure 5.5.

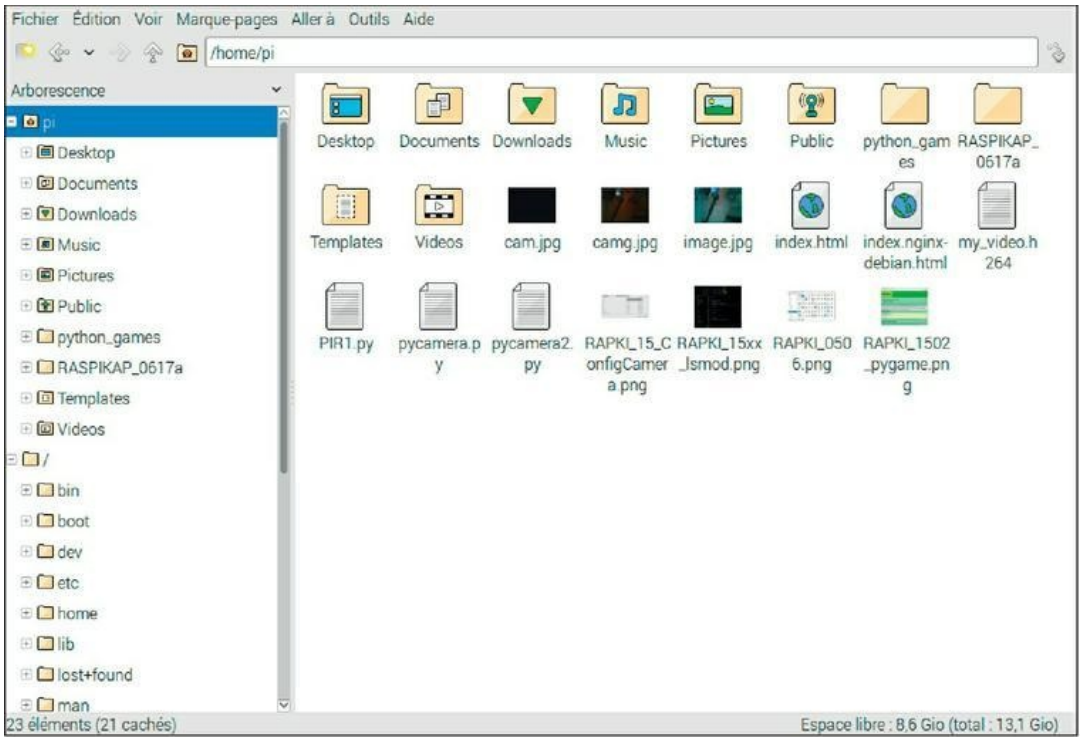


Figure 5.5 : Vue générale du Gestionnaire de fichiers.

Découvrir les fichiers dans le gestionnaire

La fenêtre du Gestionnaire de fichiers propose deux panneaux. Dans celui de gauche, tu vois la liste des dossiers que l'on appelle ici des répertoires. Pour voir le contenu d'un dossier, il suffit de cliquer son nom. Les noms des fichiers ou leurs icônes apparaissent dans le panneau droit.

Pour voir le contenu d'un sous-répertoire, tu utilises le petit triangle à gauche de son nom. Lorsque le répertoire est ouvert, le triangle pointe vers le bas. Quand il est fermé, le triangle pointe à droite.

L'ensemble des répertoires et sous-répertoires forme une structure arborescente qui part d'une racine. Le répertoire racine est la source de tous les autres.

Ce répertoire racine est si important que son nom se résume à un seul signe, la barre oblique, /. Tu noteras que sous Windows, c'est une

barre oblique inverse \, alors que sous Linux elle est penchée vers la droite.

Une énorme différence entre Linux et les systèmes Windows et macOS est qu'il n'y a pas d'unité de disque logique du style C : , D : , *etc*. Le répertoire racine est vraiment la racine de tous les fichiers, y compris ceux qui sont trouvés sur un disque amovible ou une clé USB. En général, lorsque tu connectes une clé USB ou un disque externe, ce périphérique va être raccordé à la structure arborescente au niveau d'un des sous-répertoires principaux, souvent le sous-répertoire */media/pi*. Cela correspond à son *point de montage*.

Si tu reviens à la Figure 5.5, tu vois que le répertoire racine contient un grand nombre de sous-répertoires de premier niveau. Tous les fichiers de ton système dépendent de l'un de ces sous-répertoires.

Tout en haut de la liste, tu dois voir un répertoire portant le nom *pi*, qui est le nom de l'utilisateur par défaut. C'est ton répertoire personnel, ta racine locale. À chaque nouveau compte utilisateur créé sous Linux est associé un répertoire personnel qui porte son nom. Le Gestionnaire de fichiers te propose un accès direct à ce répertoire et à ses sous-répertoires, car c'est là que tu vas travailler le plus fréquemment. Cela t'évite de devoir chercher dans la structure arborescente complète.

Faut-il en déduire qu'il y a deux répertoires personnels ? Non, il n'y en a qu'un, mais tu peux y accéder de deux manières dans le Gestionnaire de fichiers. Soit tu utilises l'accès direct en haut, soit tu navigues dans la structure complète.

Naviguer parmi les fichiers

Dans la structure arborescente du système de fichiers, à chaque fichier correspond une adresse unique qui est la séquence des différents sous-répertoires qu'il faut traverser pour atteindre le fichier.

Voici comment peut se présenter une telle adresse :

```
/home/pi/MesDocuments/etc.
```

L’adresse complète d’un fichier se nomme un *chemin d’accès*. Pour accéder à un fichier, tu dois emprunter un chemin, qui ressemble parfois à un labyrinthe. Attention au fait que les majuscules et les minuscules sont distinguées sous Linux !

Voici comment accéder à un fichier par son adresse :

1. Clique par exemple le nom du répertoire de premier niveau */home*.
2. Clique le nom du sous-répertoire */pi*.
3. Continue à cliquer jusqu’à arriver au sous-répertoire contenant ton fichier.

Pendant cette navigation de niveau en niveau, tu vois en permanence la structure arborescente dans le gestionnaire, ce qui te garantit de ne jamais être perdu.

Découvrir le nid à applications

Comme exemple concret de navigation, voyons comment accéder au

sous-répertoire dans lequel se trouvent presque toutes les applications installées dès le départ sur ta machine. C’est là que se trouve le navigateur Web dont j’ai parlé un peu plus haut dans ce chapitre. En partant de la racine, navigue dans le chemin d’accès suivant :

```
/usr/share
```

Pour naviguer, tu auras sans doute besoin d’utiliser l’ascenseur pour faire défiler la liste des noms de sous-répertoires.

La Figure 5.6 montre au format icône toutes les applications qui se trouvent dans le sous-répertoire mentionné. Cela comprend les logiciels qui sont également disponibles dans le menu général. Tu peux double-cliquer une icône pour démarrer le programme correspondant.

Figure 5.6 : Vue du sous-répertoire des applications.

Le mode super-utilisateur

Si tu as un peu navigué parmi les répertoires, tu as sans doute remarqué que tu ne parviens pas à afficher le contenu de certains répertoires. D'ailleurs, il y a un certain nombre de sous-répertoires qui te sont totalement inaccessibles !

Il y a une raison tout à fait valable. Le système d'exploitation Linux t'empêche d'accéder à des fichiers dont la moindre modification risquerait d'entraîner de sérieux dégâts au système. Puisque tu as ouvert une session comme utilisateur normal, et non comme super-utilisateur, tu n'as pas à intervenir dans les fichiers indispensables au bon fonctionnement.

Ne sois pas déçu. Tu le serais bien plus si tu rendais ton système inutilisable en un simple geste déplacé. Cela dit, il existe une porte d'accès secrète. Elle consiste à changer d'identité pour devenir super-utilisateur. Le super-utilisateur correspond au nom `root`. Il a tous les pouvoirs sur la machine.

Je décris ce mode en détail dans le Chapitre 10. Ne t'en inquiète pas pour l'instant. Dans les premières heures d'utilisation, tu n'auras besoin de basculer en mode root que pour installer un nouveau logiciel ou pour réaliser une retouche à la configuration.

Le mode super-utilisateur existe également sous macOS et sous Windows, mais il est très bien caché. Il remonte à la surface lorsque ta machine de demande de saisir ton mot de passe pour installer un logiciel. C'est un basculement temporaire en mode super-utilisateur.

Le moniteur d'utilisation

Quand tu démarres une nouvelle application, il faut à ton Raspi un minimum de temps pour mettre en place son environnement de fonctionnement. Sous Windows, tu es informé qu'il faut patienter un peu avec un sablier ou une animation circulaire. Sous macOS, c'est une roue colorée, que certains appellent la roue à pizza.

Sous Raspian, tu disposes du Moniteur d'utilisation qui est visible sous forme d'icône du côté droit de la barre des tâches, avec un pourcentage.

Cet outil permet de savoir à quel point ton Raspi est occupé. Lorsqu'il indique 0 %, c'est qu'il se croise les bras, mais à 100 %, il est totalement débordé. Mieux vaut attendre qu'il soit moins occupé avant de lancer une nouvelle opération.

Tout à fait à droite dans la barre des tâches, tu trouves l'horloge. Elle n'est pas à l'heure tant que la machine n'a encore jamais pu se connecter à Internet. Dès qu'une connexion est établie, l'horloge va se mettre à l'heure d'elle-même. Tu peux cliquer dans l'icône pour voir un calendrier.

Le menu général

Le menu général du coin supérieur gauche permet d'accéder à une sélection d'applications, et non à toutes celles qui sont installées.

N’hésite pas à parcourir tous les sous-menus pour découvrir les possibilités de ce menu général. La Figure 5.7 montre par exemple le sous-menu des accessoires.

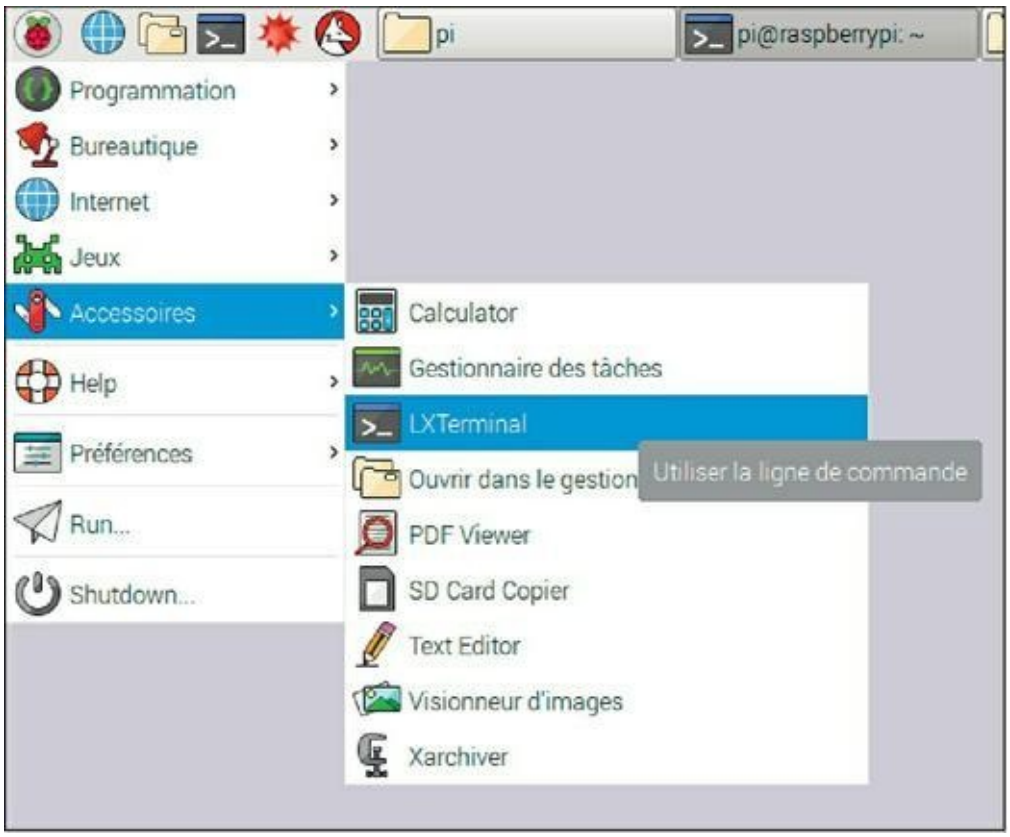


Figure 5.7 : Le menu général avec le sous-menu Accessoires.

Créer et modifier un fichier

Le système est livré avec un éditeur de texte, c’est-à-dire une sorte de traitement de texte allégé. Il te permet de créer et de modifier tes propres fichiers, mais également certains fichiers de configuration du système.

Pour démarrer cet outil, ouvre le menu général puis le sous-menu **Accessoires**. Sélectionne alors la commande **Text Editor**.

Tu vois apparaître la fenêtre vide de l’éditeur de texte. Tu peux directement saisir un peu de texte puis ouvrir le menu **Fichier** (**File**)

et choisir **Enregistrer sous** (**Save as**).

Tu vois apparaître une boîte standard de désignation de fichier (Figure 5.8). Dans cette boîte, tu sélectionnes d'abord le sous-répertoire dans lequel tu veux stocker le fichier, puis tu désignes le fichier en saisissant son nom.

Tu disposes de plusieurs options pour naviguer parmi les sous-répertoires. Tout d'abord, dans la partie supérieure, il y a plusieurs boutons qui correspondent à des répertoires déjà visités. Le panneau gauche permet de naviguer directement comme dans le Gestionnaire de fichiers. Ne cherche pas le répertoire racine /. Il a été renommé dans cette boîte Système de fichiers. C'est absolument identique.

Ne tente pas de stocker ton fichier n'importe où. Reste dans un sous-répertoire de ton répertoire personnel */home/pi*. Ce n'est qu'à cet endroit que tu as le droit de créer des fichiers.

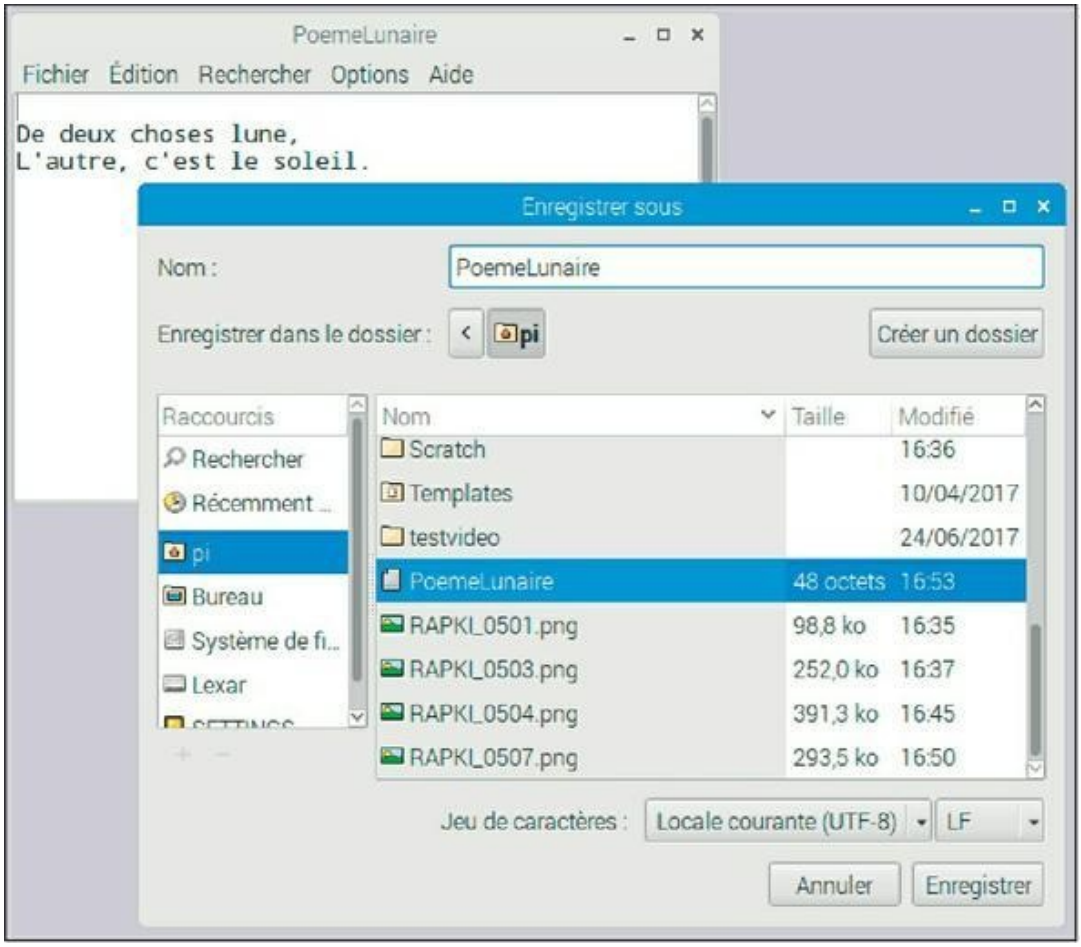


Figure 5.8 : Enregistrement d'un fichier créé avec l'éditeur de texte.

Les boutons dans la partie supérieure de la fenêtre sont vraiment pratiques : ils permettent directement d'accéder à un des sous-répertoires dont tu t'es déjà servi. Tu disposes enfin d'une liste des fichiers les plus récemment ouverts, ce qui te permet d'y revenir très facilement.

Pour un petit frisson, voyons comment charger dans l'éditeur un des fichiers du système. Ouvre le menu **Fichier** (**File**) et choisis **Ouvrir** (**Open**). Dans la boîte qui apparaît, navigue jusqu'à arriver dans ce répertoire :

```
/etc/network
```

Double-clique le nom du fichier *interfaces* ou clique une fois dans le nom ou l'icône puis clique le bouton **Ouvrir** (**Open**).

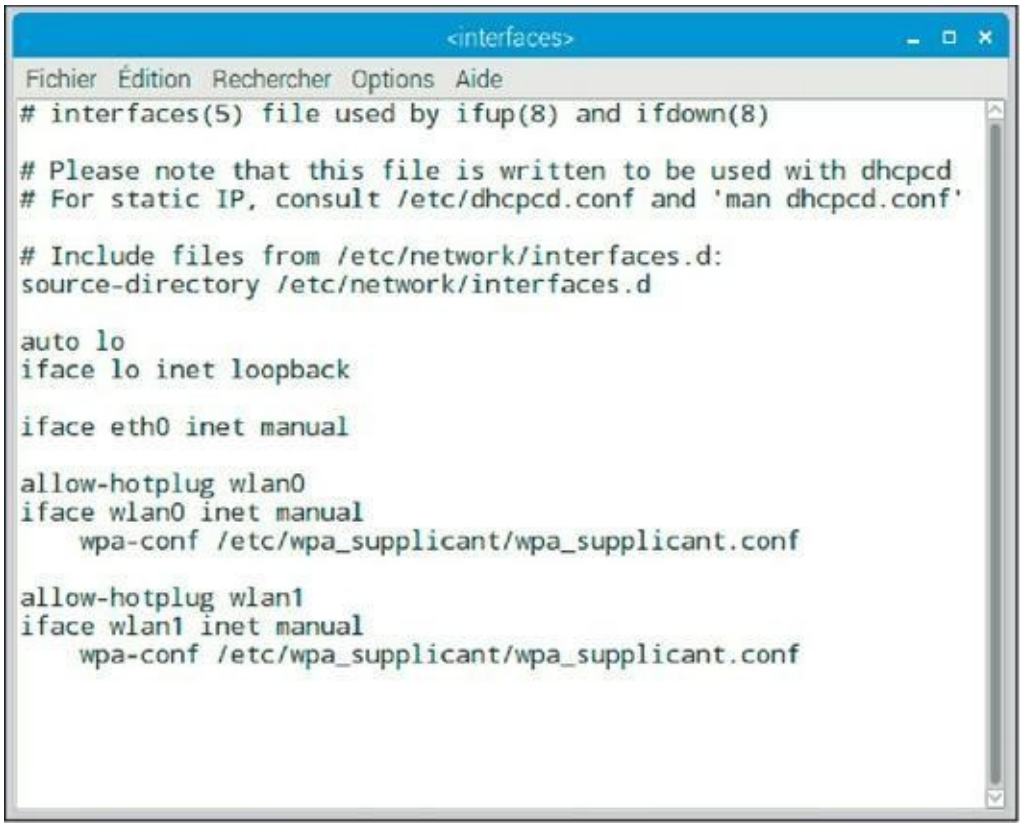


Figure 5.9 : Contenu d'un fichier de configuration système.

Le contenu du fichier n'est vraiment pas intéressant lorsque l'on ne sait pas ce qu'il signifie. Tu auras peut-être à intervenir dans ce fichier plus tard pour retoucher la configuration des connexions Internet.

Pour l'instant, inutile d'effectuer une modification, car le système t'interdira de modifier le fichier. En effet, tu as ouvert une session avec un compte d'utilisateur normal, pas de super-utilisateur. Tu peux quitter l'éditeur.

Linux réunit des centaines de fichiers, et même les professionnels ont souvent besoin de se rafraîchir la mémoire en se rendant sur des sites spécialisés avant d'intervenir sur un des fichiers du système.

Configuration de l'accès Wifi

Si tu as choisi d'utiliser une liaison Ethernet filaire (par câble), cette section ne te concerne pas pour le moment. Dans le cas contraire, voici comment procéder :

1. Ton accès Wifi est certainement protégé par un mot de passe ? Tiens-le à ta disposition.
2. Dans la barre des tâches, tout en haut à droite, tu vois le symbole d'une double liaison à gauche du symbole du haut-parleur. Si le Wifi n'est pas configuré, il y a deux croix rouges. Clique cette icône pour voir les réseaux Wifi détectés dans le voisinage. Sélectionne celui que tu veux utiliser (Figure 5.10).

Figure 5.10 : Choix d'une liaison Wifi.

3. Dans la boîte qui apparaît, saisis (ou colle) la clé ou le mot de passe pour déverrouiller la connexion Wifi (Figure 5.11).

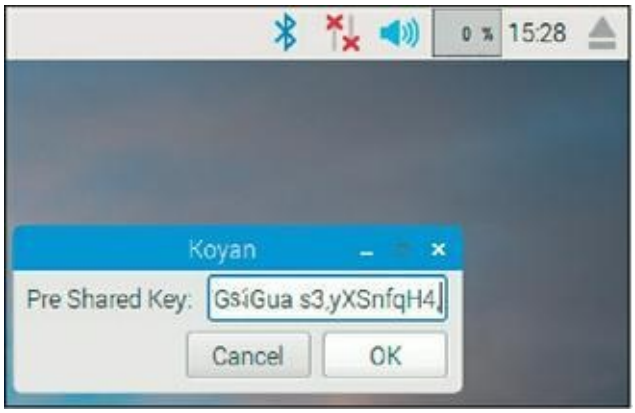


Figure 5.11 : Saisie de la clé Wifi.

À partir de ce moment, le Wifi doit fonctionner, ce que tu peux constater au changement d'aspect de l'icône de ta connexion à Internet au réseau (Figure 5.12).

Figure 5.12 : La connexion Wifi est établie.

Si ton mot de passe est trop complexe pour rester facile à saisir, une astuce consiste à stocker le mot de passe dans un petit fichier texte, sur ton PC sous Windows ou sur ton Mac, puis à placer ce fichier sur une clé USB et insérer la clé dans un port USB du Raspi. Dans l'étape de saisie du mot de passe ci-dessus, tu ouvres le fichier par double-clic pour copier/coller le code !

Dans ce cas, il faut apprendre à éjecter la clé ensuite. Lis la suite.

Disques amovibles et points de montage

Puisque tu as sans doute au moins un port USB libre, rien ne t'empêche d'y connecter un disque dur externe ou une clé USB. Cela démultiplie la capacité de stockage de ton Raspi.

1. Insère ton disque ou ta clé dans un port USB et patiente un peu.

   Une fenêtre apparaît pour confirmer que le périphérique a été reconnu et qu'il est exploitable. Il va être « monté », c'est-à-dire connecté logiquement.

Figure 5.13 : Détection d'insertion d'un disque USB.

2. Clique OK pour ouvrir le contenu dans une fenêtre du Gestionnaire de fichiers.

   Tu peux à partir de ce moment travailler avec les fichiers et notamment faire une copie de sécurité des fichiers que tu as créés sur le Raspi.

   Si tu refermes la fenêtre qui montre le disque, tu ne sais plus par où y revenir. Navigue parmi les répertoires jusqu'à celui-ci :

   ```
   /media/pi
   ```

   C'est le point de montage normal des périphériques externes.

3. Une fois que tu n'as plus besoin du disque externe, il faut l'éjecter. Clique dans le coin supérieur droit du bureau à droite de l'heure. Dans le menu, sélectionne

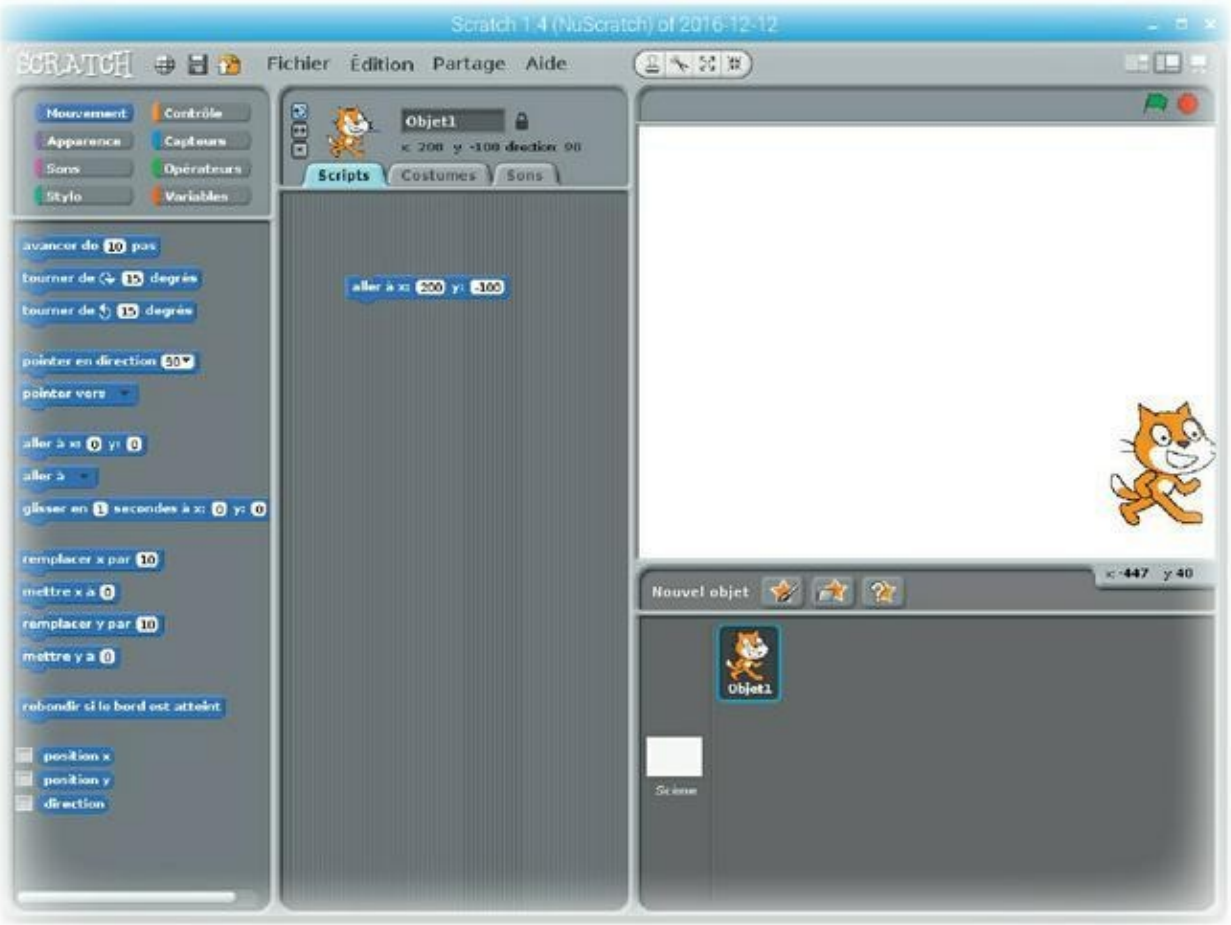
avec soin le nom du disque à déconnecter logiquement (on dit « démonter »).

Figure 5.14 : Éjection d'un disque USB.

Sois attentif au moment de choisir le disque à démonter. Si tu sélectionnes la carte système, je ne te garantis pas le résultat.

Après cette rapide découverte du système, nous allons apprendre à programmer avec ton Raspi.

Des programmes simples



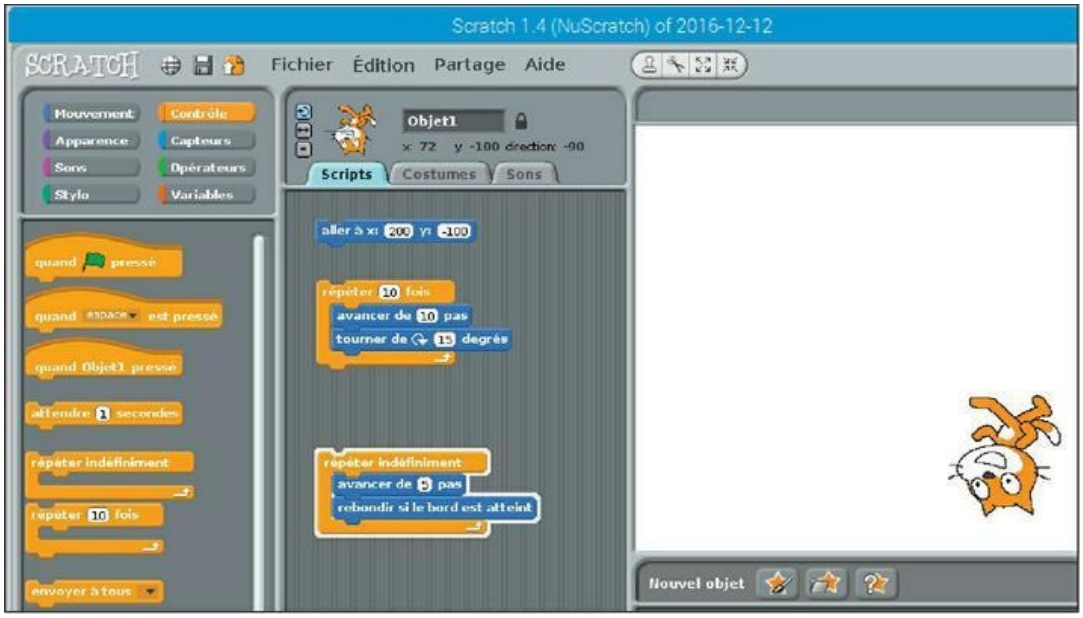
Au programme de cette partie :

Chapitre 6 : Scratcher, c’est programmer !

Chapitre 7 : Sonic Pi pour tes oreilles

Chapitre 6
Scratcher, c’est programmer !

Ton Raspberry Pi possède tout ce qu’il faut pour créer des jeux vidéo et apprendre à vraiment programmer. Dans ce chapitre, je te propose de découvrir la programmation en toute simplicité. L’outil magique qui va nous permettre d’y parvenir porte le nom Scratch. Je suis sûr qu’il te démange déjà.



Les grandes lignes de Scratch

Scratch constitue l’une des solutions les plus simples pour créer tes premiers logiciels. Normalement, programmer suppose d’écrire des mots étranges, qui ressemblent à de l’anglais simplifié, comme `write` ou `setup()`.

Avec Scratch, tu n’as presque plus besoin d’utiliser le clavier. Tu vois apparaître un grand panneau contenant de nombreux blocs portant une légende ainsi qu’une scène, c’est-à-dire l’endroit où vont se dérouler les animations.

Dans cette scène, tu vas faire apparaître des lutins que tu vas déplacer, faire rebondir contre les bords ou les uns contre les autres, et bien d’autres choses.

À chaque bloc correspond une action. Certains blocs servent à déplacer un lutin dans la scène, d’autres à le faire tourner sur lui-même ou à le faire changer de couleur. Il existe aussi des blocs pour tester si un lutin est venu au contact d’un autre ou contre un des bords de la scène.

Tu peux faire apparaître des bulles au-dessus des lutins, comme dans les bandes dessinées, augmenter ou réduire leur taille ou transformer leur aspect. Dans la figure suivante, tu peux voir un lutin qui est en train de te dire quelque chose dans une bulle.

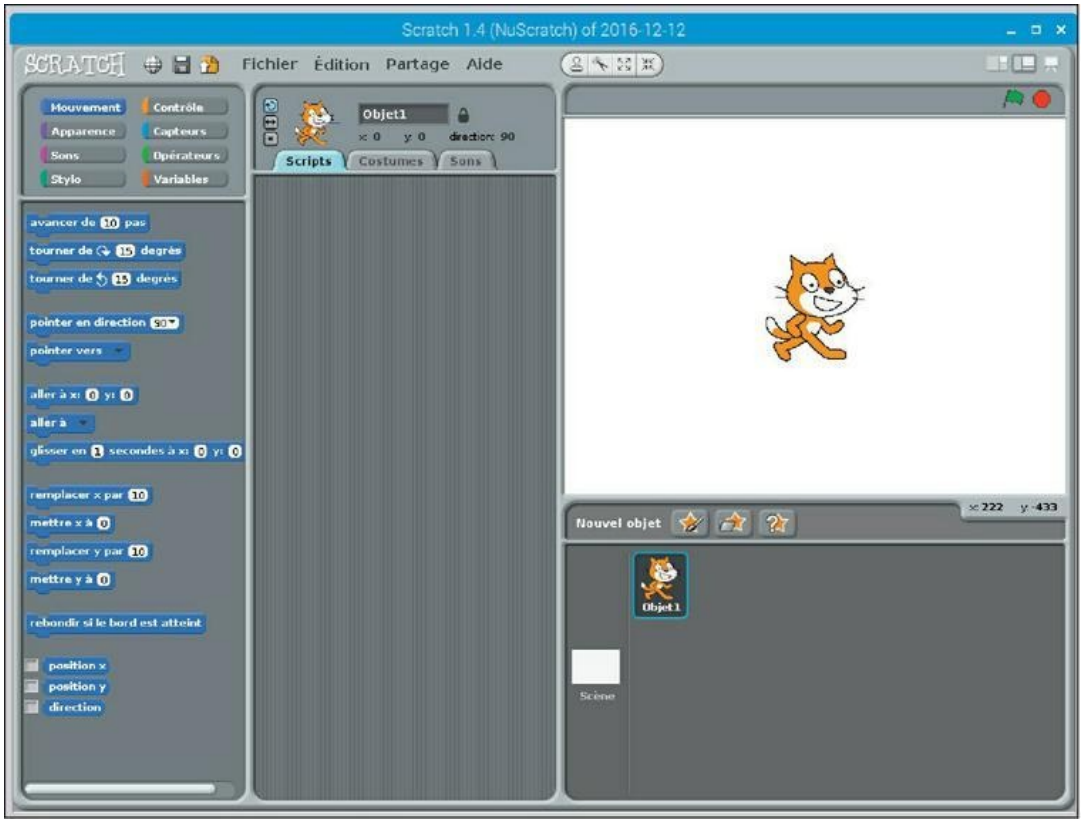


Figure 6.1 : Vue générale de Scratch.

Au départ, Scratch a été conçu pour les adolescents, mais rien n'empêche un adulte d'en tirer profit. Scratch est vraiment un outil formidable pour faire ses premiers pas en programmation, avant d'aborder un langage plus traditionnel tel que le Python.

Un langage informatique ou infolangage sert à définir la séquence d'opérations que doit réaliser un ordinateur. Il existe de très nombreux infolangages ; la plupart sont très proches les uns des autres. Les mots réservés diffèrent d'un langage à l'autre, un peu comme le français diffère de l'anglais. Dans Scratch, les blocs portent des noms d'actions simples d'emploi, et tu n'as pas besoin de les mémoriser pour pouvoir programmer.

Des blocs imbriqués

Tu peux créer un jeu vidéo ou une animation simplement en imbriquant les uns après les autres des blocs qui correspondent à autant d'actions. Lorsque tu approches un nouveau bloc sous un autre déjà présent dans la zone de script, il vient s'y accrocher.

Une série de blocs accrochés les uns à la suite des autres constitue un script. Il suffit de cliquer un des blocs pour lancer l'exécution de ce script bloc après bloc. À chaque bloc correspond normalement une action qui va être faite par un des lutins ou objets que tu as déposés dans la scène.

Les premiers blocs que tu vas utiliser servent à déplacer un lutin et à le faire tourner sur lui-même, en suivant l'ordre des blocs. Tu disposes également de blocs de répétition pour faire répéter plusieurs blocs que tu auras placés à l'intérieur de la fourche du bloc de répétition. Tu peux enfin définir des nombres et des phrases que tu stockes dans le script et faire des opérations de calcul.

Tu peux mettre en place plusieurs lutins dans la même scène et tu peux même contrôler les coulisses de la scène, en modifiant par exemple l'image d'arrière-plan.

Scratch en ligne

Il existe dorénavant une version de Scratch utilisable directement dans un navigateur Web, ce qui épargne de devoir installer l'application. Il suffit de se rendre sur le site Web de Scratch, à l'adresse `scratch.mit.edu` et de lancer la version en ligne. Cela dit, la version autonome de Scratch est installée d'office dans le système de ton Raspberry. Tu peux ainsi partir à la découverte de la programmation, même lorsqu'il n'y a pas de connexion Internet. De plus, en programmant avec ton Raspberry, tu ne vas pas monopoliser un des autres ordinateurs de la maison.

Démarrer Scratch

Scratch est bien sûr un programme à interface graphique, comme quasiment toutes les applications que l'on utilise de nos jours. Pour le démarrer, il suffit de trouver l'entrée correspondante dans le menu général.

1. Si ton Raspberry a démarré en mode texte, saisis la commande startx et valide par Entrée pour démarrer l'interface graphique.

   La Figure 6.2 montre le sous-menu permettant de démarrer Scratch.

Figure 6.2 : Menu général du Raspberry.

2. Ouvre le menu général, symbolisé par une framboise dans le coin supérieur gauche.
3. Ouvre le sous-menu `Programmation`.
4. Dans le sous-menu, sélectionne l'entrée portant le nom `Scratch`.
5. Patiente quelques instants, le temps que l'application démarre.

Dans certaines variantes du système, les options ne sont pas dans les mêmes menus que dans la version classique. Si tu ne trouves pas avec les conseils précédents, cherche un sous-menu `Programming` ou `Programmation`.

La fenêtre principale de Scratch

Comme tu peux le voir dans la Figure 6.2 un peu plus haut, la fenêtre de Scratch est constituée de plusieurs panneaux:

- **Le panneau des blocs tout à fait à gauche.** Ce panneau comporte huit catégories dans la partie supérieure et une grande zone servant de réservoir. Cette zone montre les blocs de la catégorie sélectionnée. Tu constates que les catégories ont des couleurs différentes.
- **Le panneau central avec ses trois onglets Script, Costume et Son.** C'est dans la partie inférieure de ce panneau que tu déposes les blocs depuis le panneau de gauche. Le deuxième onglet permet de créer et de modifier l'aspect de l'objet sélectionné (le lutin) en lui associant plusieurs costumes. Le troisième onglet permet de définir des sons qui seront associés aux lutins.
- **Le panneau supérieur droit de la scène.** Dans cette grande zone blanche, tu trouves dès le départ un lutin, qui est le chat, mascotte de Scratch. C'est dans cette scène que va se dérouler l'animation que tu as programmée.
- **Le panneau des lutins ou des objets.** Ce panneau contient tous les lutins dont tu vas te servir dans ton projet, que tu les aies importés ou créés de toutes pièces. C'est également ici que tu peux accéder aux coulisses de la scène.

Dès le démarrage de Scratch, il y a un lutin dans la scène : le chat. Tu vas pouvoir modifier son aspect en intervenant dans la page Costumes et lui faire faire des cabrioles en utilisant les blocs de mouvement.

Le principe de la scène

Pour déplacer un lutin dans la scène, tu ne vas pas utiliser les quatre sens habituels haut, bas, gauche et droite. Tu utilises des coordonnées. Ce sont des nombres qui déterminent la position de l'objet par rapport au centre de la scène. Ces coordonnées correspondent aux valeurs x et y, comme en géométrie. La valeur x détermine la position dans le sens horizontal (de la gauche vers la droite) et la valeur y la détermine dans le sens vertical, de bas en haut.

Au départ, le chat est pile au centre de la scène. Si tu regardes ses coordonnées, tu vois qu'elles sont toutes les deux égales à zéro. Pour déplacer l'objet du côté droit, il suffit de donner une valeur supérieure à zéro à x. Pour le faire aller à gauche, il faut lui donner une valeur négative pour x.

Lorsque la coordonnée x vaut 100, l'objet est dans la partie droite de la scène et quand elle vaut -100, il est dans la partie gauche.

La même logique s'applique à y. Quand y vaut 100, l'objet est au-dessus du milieu et quand y vaut -100, l'objet ou le lutin est en dessous.

Les deux coordonnées XY sont totalement indépendantes l'une de l'autre. Tu peux ainsi faire glisser un objet vers la droite ou vers la gauche sans le faire changer de hauteur, de même dans le sens vertical.

Pour obtenir un déplacement en diagonale, il faut bien sûr changer en même temps les deux coordonnées.

Le Tableau 6.1 donne quelques exemples.

Tableau 6.1 : Exemple de déplacements dans la scène.

Valeur de Y	*Position du lutin*
x est positif, par exemple 100	Partie droite de la scène

x est négatif, par exemple -100	Partie gauche de la scène
y est positif	Moitié supérieure de la scène
y est négatif	Moitié inférieure de la scène
x vaut zéro	Centre dans le sens horizontal
y vaut zéro	Centre dans le sens vertical
x et y valent zéro	Exactement au centre de la scène

Pourquoi les concepteurs de Scratch ont-ils choisi d'utiliser des coordonnées numériques au lieu d'utiliser la gauche, la droite, le haut et le bas ? C'est tout simplement parce qu'il est beaucoup plus simple de travailler avec des valeurs numériques en programmation. Cela permet de positionner précisément un objet, comme tu vas bientôt le voir.

Déplacer un premier lutin

Voyons tout de suite comment déplacer notre lutin dans la scène. Nous allons utiliser un bloc de mouvement, `Aller à x: y:` .

1. Pour être certain que le script que tu vas créer va bien être associé au chat, clique dans le panneau inférieur droit des objets une seule fois dans l'icône du chat.
2. Si nécessaire, clique une fois en haut du panneau gauche dans la catégorie `Mouvement`.
3. Dans la liste des blocs à gauche, repère le bloc contenant la mention `Aller à x: y:`.

Agrippe ce bloc et fais-le glisser pour le déposer dans la zone vide du centre, dans le panneau Script, puis relâche.

Lorsque Scratch va exécuter ce bloc d'action, il va positionner l'objet concerné à l'endroit qui est défini par les valeurs indiquées à la suite des deux noms x: et y: dans le bloc. Si tu n'as rien changé, les deux valeurs sont égales à zéro. Le bloc possède normalement l'aspect suivant :

```
Aller à X:  0 Y:  0
```

4. Clique dans la zone de saisie juste à droite de l'indication `x:` dans le bloc. Dès que le contenu est sélectionné, saisis la valeur `200` puis frappe la touche Tab à gauche du clavier (juste au-dessus du cadenas).
5. Pour vérifier que le bloc fonctionne, clique une fois dans le bloc, ce qui lance son exécution. Tu constates que le chat se déplace vers la droite dans la scène. C'est tout simple non ?
6. Fais de même pour modifier la coordonnée Y en indiquant la valeur `-100`.
7. Clique une fois dans le même bloc pour l'exécuter. Tu remarques que le chat s'est encore déplacé vers la droite, mais également vers le bas.

   Tu dois obtenir à peu près la même chose que le contenu de la Figure 6.3.

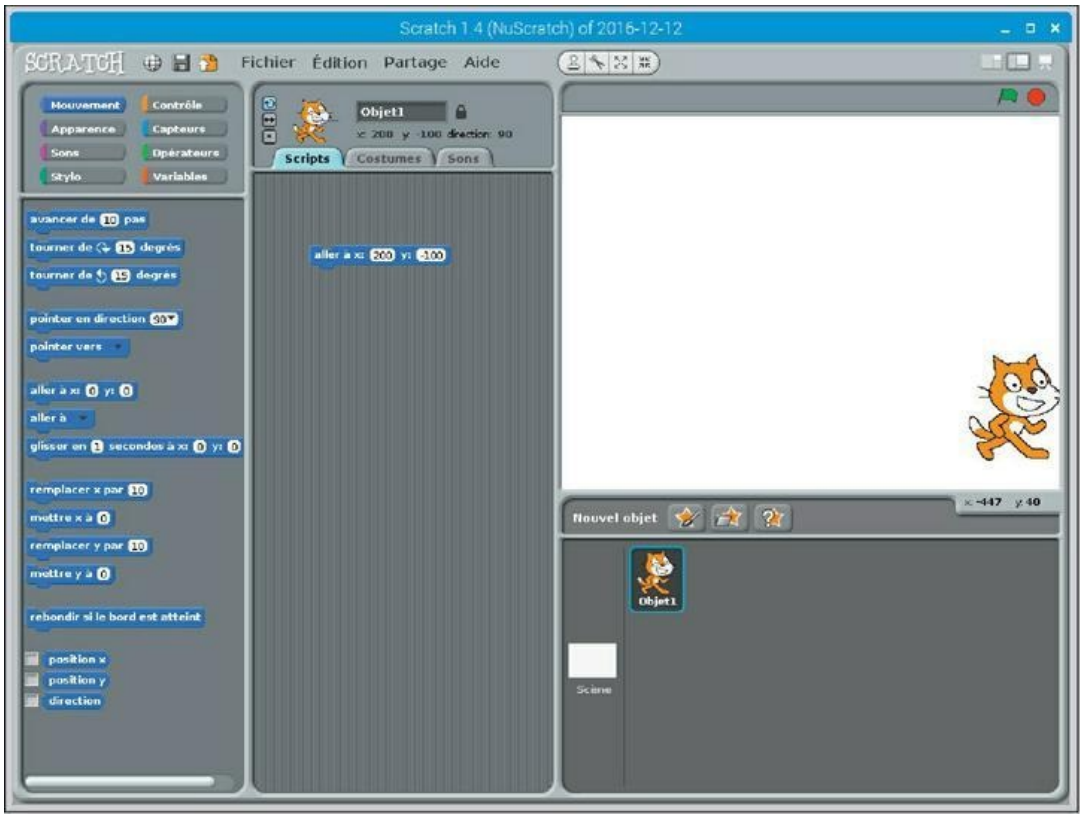


Figure 6.3 : Résultat de l'exécution du premier bloc de mouvement.

Pas de panique si ton lutin ne se trouve pas au même endroit que dans la figure précédente. Tout dépend de la taille de la scène, et celle-ci dépend à son tour de la résolution de ton écran. Pour connaître les limites dans les quatre directions, utilise l'astuce suivante : déplace le pointeur de souris vers chacun des quatre bords sans cliquer. Lis à chaque fois les valeurs des coordonnés qui s'affichent en bas à droite de la scène.

Pour centrer un lutin

Est-ce que tu devines comment utiliser le bloc `Aller à x: y:` pour ramener ton lutin en plein milieu de la scène ? Tu peux te servir du contenu du Tableau 1.1.

Bien sûr, il suffit de donner à XY la valeur zéro pour ramener le lutin au centre.

Essaye maintenant de saisir d'autres valeurs pour les coordonnées afin de voir comment tu peux déplacer l'objet. Au bout d'un certain moment, tu devines où l'objet va se retrouver quand tu vas exécuter le bloc, avant même d'essayer.

Dans la liste des blocs à gauche, il y a deux couples de blocs apparentés à celui que nous venons d'utiliser :

Remplacer X par 10

Mettre X à zéro

Remplacer Y par 10

Mettre Y à zéro

La danse du Moonwalk

Dans la vraie vie, les êtres ne se déplacent pas instantanément d'une position à l'autre. Pour que les mouvements soient plus réalistes, il suffit d'utiliser le bloc `Glisser`.

Il fonctionne de la même manière que le bloc `Aller à x: y:` , mais son nom laisse deviner que tu peux rendre le déplacement progressif. Teste ce bloc en donnant une valeur un peu plus grande au temps de glissement afin de juger du résultat.

Orienter un lutin

Pour l'instant, notre chat regarde vers la droite. Tu peux modifier son orientation avec d'autres blocs puis lui dire d'avancer dans la direction vers laquelle il regarde.

Tu disposes à cet effet des blocs `Avancer de`, `Tourner (flèche) de` et `S'orienter à` (ou Pointer en direction). Ces blocs se trouvent tout en haut de la liste des blocs de mouvement. Tu peux les tester en modifiant la valeur pour découvrir leur fonctionnement.

Pour changer l'orientation du lutin, tu disposes du bloc `S'orienter à` (ou Pointer en direction). La valeur indiquée correspond à 1 degré. Si tu saisis `360`, le lutin va faire un tour sur lui-même, ce que tu ne verras même pas parce que cela se produira trop vite. Si tu indiques `180`, le lutin va faire un demi-tour et avec la valeur `90`, il va tourner d'un quart de tour.

Tu remarques que dans la zone de saisie de la valeur, tu peux non seulement indiquer une valeur en la tapant au clavier, mais également sélectionner une direction prédéfinie. Prends le temps de comprendre à quelle direction (la gauche, la droite, le haut et le bas) correspondent les quatre valeurs.

Orientation réelle et apparente

Tu peux toujours changer l'orientation d'un lutin dans la scène, ce qui va déterminer dans quel sens il va se déplacer lorsque tu vas lui demander d'avancer. Pour autant, il n'est pas toujours intéressant visuellement de faire tourner l'image du lutin dans le sens choisi. Cela peut sembler confus, mais tu vas tout comprendre en quelques instants.

En effet, Scratch propose des options pour déterminer comment va apparaître le lutin suite à des blocs d'orientation ou de rotation.

Observe bien le haut du panneau central des scripts. À gauche de l'image du lutin, tu vois trois petits boutons.

Tout d'abord, sélectionne ton lutin dans le panneau inférieur droit puis clique le petit bouton `i` pour accéder aux détails.

Clique un des boutons pour savoir ce qu'il permet de faire. Voici l'explication des trois boutons en commençant par le haut :

» `Style de rotation :` avec cette option, le lutin tourne toujours comme demandé, c'est-à-dire vers le haut, le bas, la gauche ou la droite. Autrement dit, l'image du lutin peut se retrouver tête en bas.

» `Gauche ou droite seulement :` le lutin ne peut que regarder vers la droite ou vers la gauche, même si tu lui demandes une orientation vers le haut ou le bas.

» `Pas de rotation :` le lutin ne change jamais d'orientation visuelle. Cela n'empêche pas de modifier la direction de déplacement, mais l'apparence ne change jamais.

Créer un script simple

Pour créer un script, il suffit d'enchaîner plusieurs actions, c'est-à-dire plusieurs blocs. Commençons par un script constitué de deux blocs.

1. Dans le réservoir gauche, prends le bloc `Avancer de` et fais-le glisser pour le déposer dans la zone des scripts au centre, à l'écart d'un bloc qui s'y trouve déjà.

2. Prends dans les blocs à gauche le bloc portant le nom `Tourner de 15°` et fais-le glisser sans relâcher la souris pour qu'il se positionne juste sous le bloc précédent. Ne relâche pas le bloc.

   Tu dois voir apparaître une ligne blanche épaisse horizontale (Figure 6.4).

Figure 6.4 : Ajout d'un bloc à la suite d'un autre.

3. Tu peux maintenant relâcher la souris.

   Normalement, le second bloc vient s'accrocher sous le précédent par son encoche.

Tu as créé ton premier script !

Il suffit de cliquer sur n'importe lequel des deux blocs de ce script pour en provoquer l'exécution. Le lutin va se déplacer puis tourner.

Ce premier script ne possède que deux blocs, mais le principe reste le même lorsque tu crées un script avec des dizaines de blocs. Scratch commence par le bloc du haut puis exécute un bloc après l'autre en descendant.

Le fait de réaliser les actions des blocs correspond à l'exécution du script. Tu peux imaginer un sorcier caché derrière qui lance les actions l'une après l'autre. En réalité, dans le script, le sorcier, c'est le processeur de ton ordinateur.

La plupart des blocs possèdent également une encoche dans leur partie supérieure, ce qui permet d'ajouter un bloc avant un autre. Dès qu'il y a une encoche ou un téton, c'est qu'une connexion est possible.

Comment diviser un script

Parfois, tu auras besoin de démonter ou de diviser un script, par exemple pour enlever le dernier bloc. Ou bien, il te faudra ouvrir le script pour ajouter une nouvelle partie en plein milieu. Voici comment diviser un script :

1. Clique dans le bloc et maintiens l'appui pour pouvoir le faire glisser.

   Le bloc se détache et Scratch affiche la fameuse ligne blanche.

2. Fais glisser le bloc suffisamment pour qu'il soit éloigné des autres, dans une zone libre.

   Dès que la ligne blanche disparaît, tu peux relâcher. Le bloc ou les blocs sont dorénavant séparés du reste.

Le menu local de bloc

Tu peux cliquer avec le bouton droit dans un bloc pour ouvrir un menu local qui offre plusieurs commandes :

- `Aide :` permet d'obtenir de l'aide au sujet de l'usage du bloc, dans une petite fenêtre. Tu la refermes en cliquant OK.
- `Dupliquer :` cette commande sert à faire une copie du bloc ou du script entier. La copie va

apparaître dans la même zone des scripts.

» `Supprimer :` permet de supprimer le ou les blocs. Si tu supprimes un bloc par erreur, tu peux utiliser la première commande du menu Édition qui s'intitule Restaurer ou Ne pas effacer. Le menu général se trouve tout en haut de la fenêtre de Scratch.

Un script pour repartir au centre

C'est maintenant à toi de créer seul un petit script qui va replacer le lutin au centre de la scène et le réorienter dans sa direction de départ. Il faut donc que le lutin regarde vers la droite.

Il suffit de mettre en place les blocs que nous avons déjà rencontrés et de saisir les valeurs appropriées pour les coordonnées, ici ce sera zéro et zéro. Tu constates que tu peux mettre en place plusieurs scripts dans la zone des scripts. L'essentiel c'est qu'ils ne se touchent pas. Pour en lancer un, il suffit de cliquer dessus.

Des blocs pour contrôler

Tu auras souvent besoin de faire répéter une action plusieurs fois dans un script. Nous allons repartir du script qui permettait d'avancer et de tourner un peu plus haut. Si tu l'avais divisé, recolle les deux blocs et clique plusieurs fois pour voir s'enchaîner le déplacement et la rotation.

Au lieu de cliquer sans arrêt, tu peux utiliser un bloc pour faire répéter les deux opérations.

La première technique à laquelle tu peux penser pour faire répéter les deux actions consiste à dupliquer le script puis à coller les copies à la suite de l'original.

Cela pourra suffire lorsqu'il y a quelques blocs d'actions à répéter, mais cela ne convient pas pour un plus grand nombre de répétitions,

ni lorsque le bloc d'actions est très grand.

Scratch offre une solution bien plus efficace. Ouvre la catégorie de blocs intitulée `Contrôle` tout en haut du panneau gauche. C'est la catégorie orange. Tu vois apparaître une nouvelle série de blocs. Ce sont les blocs de contrôle. C'est grâce à eux que tu vas vraiment commencer à faire de la programmation.

Voici le genre d'opération que tu vas pouvoir faire avec ces blocs :

- Répéter indéfiniment une série de blocs.
- Répéter un certain nombre de fois une série de blocs puis poursuivre.
- Faire démarrer le script par la frappe d'une touche du clavier.
- Mettre le script en pause pendant un certain nombre de secondes.
- Mettre un script en pause jusqu'à ce qu'un événement survienne.
- Répéter un script jusqu'à ce que quelque chose survienne.
- Tester une valeur numérique, la position d'un lutin ou une autre condition avec un bloc Si Alors.
- Forcer l'arrêt du script ou de tous les scripts.

Exploiter des blocs de contrôle

Les blocs de la catégorie `Contrôle` peuvent être placés à trois endroits dans un script :

- tout au début du script ;

- tout à la fin du script ;
- autour de plusieurs blocs.

Les blocs de démarrage se distinguent des autres par leur partie supérieure arrondie. Cela signifie que tu ne peux pas accrocher un bloc par-dessus. Ils servent à démarrer l'exécution du script lorsqu'un événement se produit. Le script ne démarrera que quand l'événement sera détecté.

Par exemple, le bloc portant le nom `Quand espace est pressé` fait démarrer le script dès que tu frappes la touche Espace. Le nom de la touche fait partie d'une liste déroulante, ce qui te permet d'en choisir une autre.

Les blocs de fin d'exécution sont sans encoche dessous. Cela signifie que tu ne peux pas ajouter de bloc après. Ils servent à arrêter l'exécution du script.

Le troisième type de bloc ressemble à une fourche. L'intérieur de la fourche est destiné à recevoir un ou plusieurs blocs.

Pour ajouter un tel bloc, tu auras souvent besoin de diviser le script comme tu l'as vu un peu plus haut. Les blocs qui doivent entrer dans la fourche sont ensuite repositionnés une fois le bloc de contrôle mis en place.

Fais par exemple un essai avec le bloc de répétition qui s'intitule `Répéter 10 fois`. Dépose-le dans la zone des scripts de telle manière qu'il englobe les deux blocs de ton script. Tu remarques que la branche basse de la fourche descend pour accueillir les deux blocs.

Tu peux voir le résultat dans la Figure 6.5. Si tu lances l'exécution en cliquant le bloc, tu vois avancer et tourner le lutin 10 fois. Tu peux saisir une autre valeur pour la répétition.

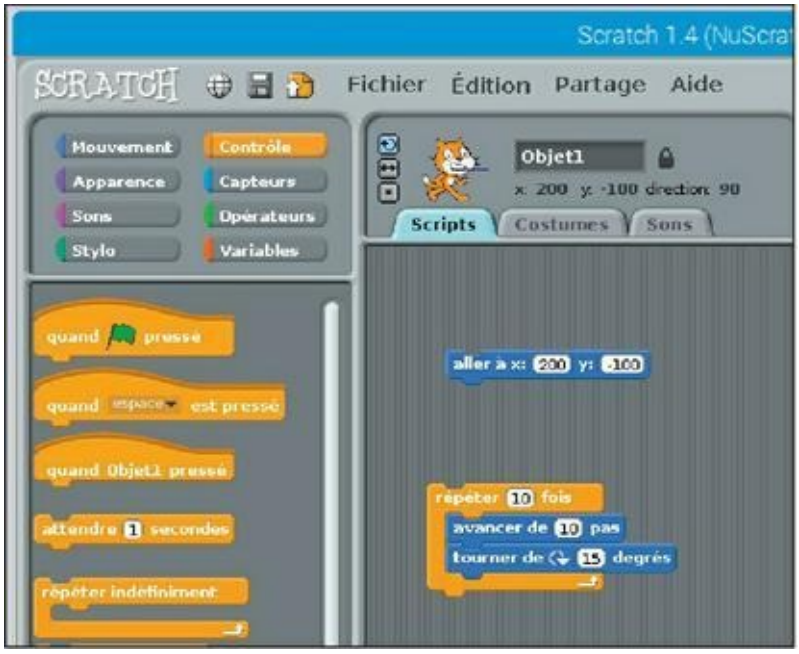


Figure 6.5 : Un script avec un bloc de répétition.

Si le lutin ne tourne pas, vérifie quelle option de rotation est active dans le haut de la fenêtre de script, juste à gauche de l'image du lutin. J'ai expliqué à quoi servaient ces trois boutons un peu plus haut dans ce chapitre.

Pour arrêter un script

Dans le nombre de tours de répétitions du bloc de contrôle, saisis par exemple la valeur `100` puis frappe la touche Entrée. Lance ensuite l'exécution du script.

Dorénavant, le lutin va se déplacer sans cesse, ou tout du moins pendant un bon moment.

Tu peux arrêter l'exécution avant d'être arrivé à la fin du décompte. Utilise par exemple le bouton rouge en haut à droite de la scène. C'est celui juste à côté du drapeau vert, qui sert à démarrer l'exécution de tous les scripts.

Tu peux aussi cliquer sur un des blocs du script pour l'arrêter. Tu sais que le script est en cours d'exécution quand il y a une bordure

blanche autour.

Savoir rebondir

Voyons comment écrire un script qui fait rebondir le lutin quand il touche l'un des quatre bords de la scène. Il y a deux solutions, une simple et l'autre plus compliquée.

La solution simple consiste à utiliser dans la catégorie `Mouvement`, le bloc intitulé `Rebondir si le bord est atteint`. Prépare un nouveau petit script en associant un bloc `Aller à` et un bloc `Rebondir si le bord est atteint`. Insère les deux blocs dans un bloc de contrôle `Répéter indéfiniment`.

Si tu as conservé dans un coin des scripts celui permettant de replacer le lutin au centre, clique-le pour ramener le lutin au milieu. Sinon, ramène-le à la main en faisant glisser le chat.

La Figure 6.6 montre le résultat dans la zone des scripts.

Tu peux aussi activer l'option qui limite l'orientation du lutin aux mouvements latéraux (le deuxième petit bouton à gauche de l'icône du lutin).

Tu peux maintenant lancer l'exécution du script. Le lutin doit se promener de gauche à droite sans jamais arrêter. Clique à nouveau dans un bloc pour arrêter l'exécution.

Que se passe-t-il si tu fais tourner le lutin d'abord ? Utilise un bloc de rotation. Si l'option de rotation interdite est activée, tu ne verras pas de différence, mais le lutin rebondit maintenant dans tous les sens. Clique à nouveau dans le script pour l'arrêter.

Découvrons maintenant la technique moins simple, mais beaucoup plus instructive.

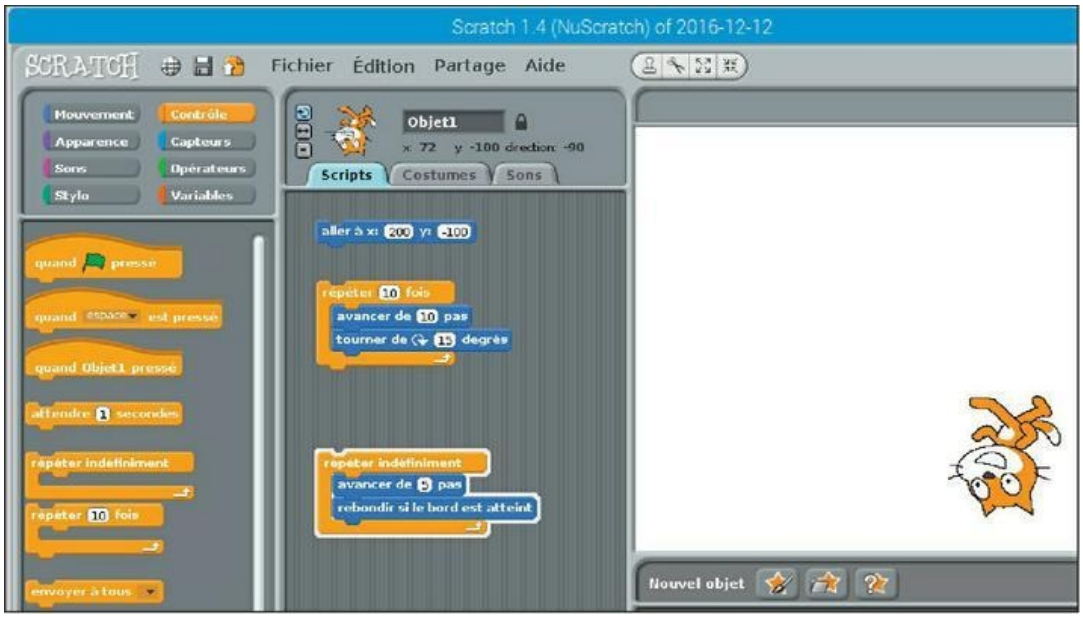


Figure 6.6 : Un script pour mieux rebondir.

Ta première variable

Qu'est-ce que cela signifie de refaire rebondir sur le bord de scène ? Que se passe-t-il vraiment ?

Rebondir suppose de faire faire demi-tour à l'objet. Tu peux donc utiliser un bloc de rotation pour simuler le robot. Il suffit d'orienter le lutin vers la gauche lorsqu'il touche le bord droit et vers la droite lorsqu'il touche le bord gauche.

Mais tu peux vouloir faire faire autre chose à ton lutin. Tu peux vouloir le faire sauter lorsque tu frappes une touche, ou le faire se cacher derrière un autre lutin, ou même le faire rebondir avant qu'il touche un bord.

Pour toutes ces actions, il faut à tout moment que tu saches quelles sont les coordonnées actuelles du lutin. Cela te permet ensuite de modifier ses coordonnées.

La solution bête et très fatigante consiste à prévoir des dizaines de blocs `Aller à` avec des valeurs différentes pour x et y afin de

gérer toutes les positions que peut occuper le lutin dans la scène. C'est impraticable.

La bonne solution consiste à mémoriser les valeurs des coordonnées pour pouvoir les modifier comme bon te semble. Dans Scratch, il suffit de créer des variables.

Une variable est un emplacement en mémoire qui contient une valeur et qui est symbolisé par un nom. Le bloc doit porter un nom unique pour pouvoir le distinguer des autres blocs de variable. Son contenu permet de stocker une valeur numérique.

Une variable peut également contenir une lettre de l'alphabet, un mot ou même une phrase.

La puissance des variables

Scratch permet de faire trois choses superbes avec des variables. Tout d'abord, tu peux les créer grâce aux blocs dédiés à cet usage. Dès que tu crées une variable, une série de blocs liés à cette variable apparaît. Tu peux stocker une valeur dans la variable ou y inscrire par exemple le résultat d'une addition.

Dès que tu crées la variable, elle apparaît dans le coin supérieur gauche de la scène. Lorsque tu ne veux pas que la variable reste visible, il suffit d'utiliser le bloc `Cacher la variable`. Pour la rendre visible à nouveau, tu ajoutes le bloc `Montrer la variable`.

Les variables permettent de réaliser des opérations mathématiques. Tu peux faire une addition, une soustraction, une multiplication et une division entre la valeur que contient la variable et une autre valeur. Tu peux même combiner le tout !

Encore plus fantastique : tu peux utiliser une variable partout où tu vois une valeur numérique dans les blocs. Tu peux par exemple utiliser une variable pour la valeur du bloc de déplacement `Aller à`. Le déplacement que va provoquer ce bloc lorsqu'il sera exécuté correspondra à la valeur qui aura été lue dans la variable.

Les variables ouvrent un champ de possibilités énormes, en permettant à ton script de ne pas toujours se comporter de la même manière. Tu peux changer la valeur d'une variable à la main ou par calcul ou en faisant dépendre cette valeur d'une autre valeur, comme la position d'un autre lutin.

Créer notre première variable

Pour créer une variable, il suffit d'ouvrir d'abord la catégorie correspondante parmi les catégories de blocs. Il s'agit de la catégorie orange qui s'appelle `Données`. Tu dois voir trois possibilités s'offrir à toi :

- créer une variable ;
- supprimer une variable ;
- créer une liste.

Une liste est un genre de variable un peu complexe que nous ne verrons pas dans ce livre. C'est en fait une variable qui en contient d'autres. Chacune des sous-variables porte un numéro, ce qui permet d'accéder à chacune en indiquant ce numéro (c'est un indice).

Voici comment créer une variable :

1. Affiche les blocs de la catégorie `Données` et clique `Créer une variable` (ou `Nouvelle variable`).

   Tu vois apparaître une fenêtre comme celle de la Figure 6.7.

Figure 6.7 : La boîte de création d'une variable.

2. L'essentiel pour créer une variable consiste à choisir un nom qui n'est pas encore utilisé. Dans cet exemple, choisis par exemple de saisir le nom `lutino` dans la zone du nom de la variable.

3. Il y a deux options sous la zone de saisie. Laisse active l'option `Pour tous les lutins`. Tu peux valider la saisie par OK.

   Cette confirmation entraîne plusieurs événements. Tout d'abord, Scratch va créer plusieurs nouveaux blocs. Dans le coin supérieur gauche de la scène, tu dois voir apparaître une boîte avec le nom de la variable et une valeur numérique.

Lors de la création de la variable, la valeur est toujours égale à zéro. Tu peux voir le résultat dans la Figure 6. 8.

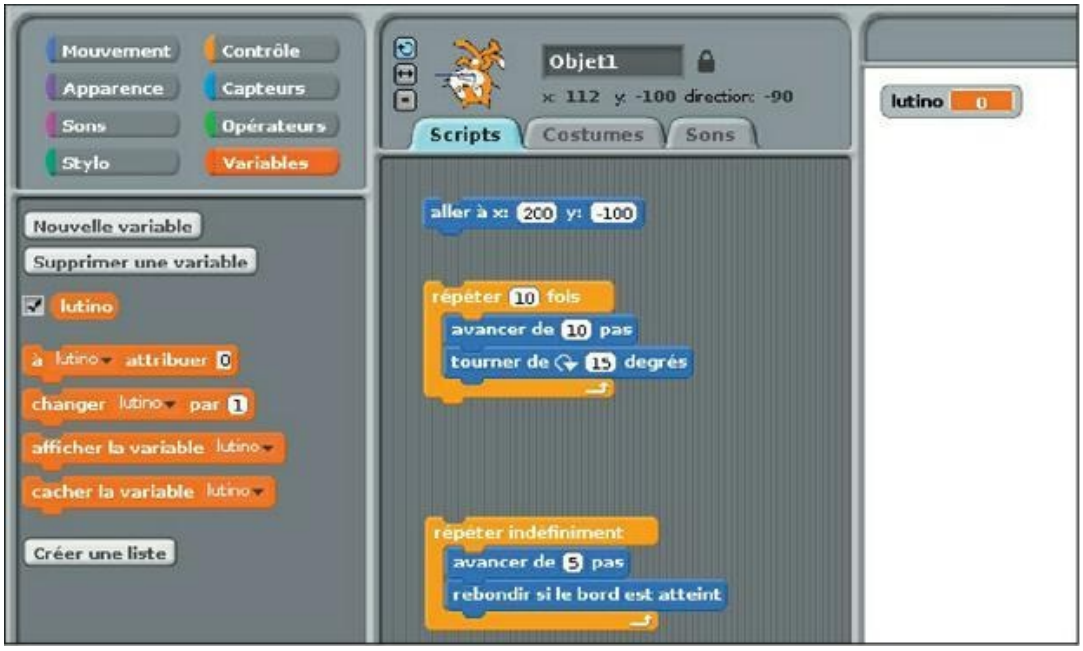


Figure 6.8 : La première variable a été créée.

J'ai dit plus haut qu'une variable pouvait remplacer n'importe quelle valeur numérique. Tu peux ainsi utiliser le bloc `Mettre XX à zéro` pour changer la valeur d'une variable à partir de celle d'une autre. Tu peux également utiliser le bloc `Ajouter à` pour écrire une nouvelle valeur dans la variable. Les scripts deviennent vraiment polyvalents quand on utilise cette possibilité d'influer sur la valeur de certaines variables par la valeur d'autres variables, d'un lutin à l'autre. Tu entres alors vraiment dans le monde de la programmation.

Variable globale ou locale

Lors de la création d'une variable (la déclaration), tu as le choix entre deux possibilités avec les deux boutons radio. Lorsque tu choisis l'option `Pour ce lutin uniquement`, la variable n'est utilisable qu'avec le lutin qui est actif au moment où tu crées la variable. La valeur est inaccessible depuis les autres lutins, et donc

non modifiable. Cette privatisation de variable est très utile dans certains cas. Cela permet par exemple d'utiliser le même nom pour plusieurs variables, puisqu'elles ne se voient pas les unes les autres. Les variables locales sont beaucoup plus faciles à réutiliser lorsque tu récupères un certain nombre de blocs d'un script pour les réutiliser dans un autre, par exemple un autre lutin.

Dans d'autres cas, il te faudra au contraire permettre à tous les autres scripts d'accéder à la variable. Dans ce cas, tu choisis l'option `Pour tous les lutins` au moment de la création de la variable. La variable devient à partir de ce moment utilisable dans tout ton projet.

Ce choix parmi les deux options peut sembler accessoire aux débutants, alors qu'il est très important. Les programmeurs réfléchissent bien avant de décider de faire de leur prochaine variable une variable locale ou une variable globale, c'est-à-dire variable privée ou publique. Si tu as toutes les variables accessibles de partout, tu auras beaucoup plus de mal à bien contrôler le fonctionnement. Inversement, si tu verrouilles toutes les variables en les rendant privées, tu ne parviendras pas à ton objectif.

Utiliser une variable dans un bloc

Le principal intérêt des variables est qu'elles sont utilisables à la place des valeurs numériques que contiennent certains blocs. Au lieu d'utiliser une valeur fixe dans un bloc, par exemple `10`, tu peux déposer à la place le nom d'une variable, la valeur de la variable pouvant changer en fonction d'autres conditions, par exemple suite à un calcul.

Dans Scratch, l'utilisation d'une variable dans un bloc est très simple, puisqu'il suffit de faire glisser et de relâcher le nom de la variable à la place de la valeur numérique :

1. Agrippe le bloc du nom de la variable et relâche-le à l'écart dans la zone des scripts.

Prends soin de choisir le bloc qui ne contient que le nom de la variable, et pas un des autres blocs qui ont été créés suite à la création de la variable.

2. Dans la catégorie de blocs `Mouvement`, dépose à l'écart dans la zone de scripts un bloc `Aller à x: y:` .

3. Il ne reste plus qu'à faire glisser le bloc du nom de la variable dans la zone numérique du bloc de mouvement.

Tu sais que tu as réussi l'insertion lorsque tu vois le nom de la variable remplacer la valeur zéro. Tu dois obtenir quelque chose comme dans la Figure 6.9.

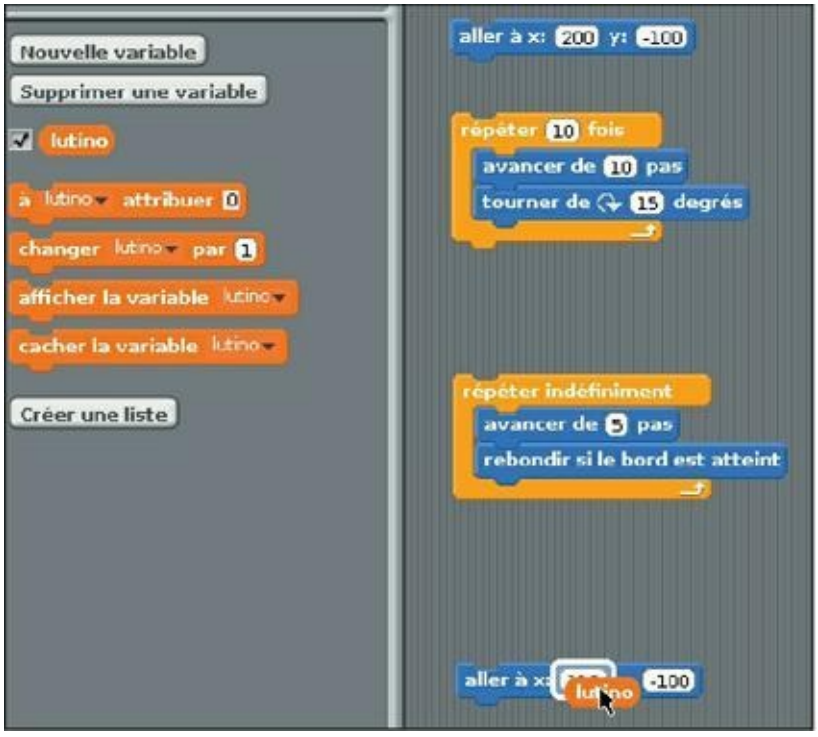


Figure 6.9 : Utilisation d'une variable dans un bloc de mouvement.

Pour faire varier la valeur de la variable

La valeur qui est affichée pour la variable est sa valeur actuelle. Affiche les blocs de la catégorie `Variables` à gauche pour voir s'afficher les blocs `Mettre à` et `Ajouter`.

Le bloc `Mettre à` permet de donner à la variable la valeur qui est indiquée dans le bloc.

Le bloc `Ajouter` ajoute à la valeur actuelle de la variable la valeur qui est indiquée dans le bloc. Au départ, la valeur est égale à un. Autrement dit, le fait d'insérer ce bloc fait augmenter la valeur de la variable de 1. Tu peux bien sûr indiquer ta valeur de changement.

Fais un essai :

1. Clique dans la zone de saisie du bloc `Mettre à` (à `lutino attribuer 0`) pour saisir la valeur `10`.
2. Clique une fois dans le même bloc pour voir ce qui se passe.
3. Clique plusieurs fois de suite dans le bloc `Ajouter` (`changer lutino par 1`).

L'affichage de la variable dans le coin supérieur gauche de la scène change en conséquence, comme le montre la Figure 6.10.

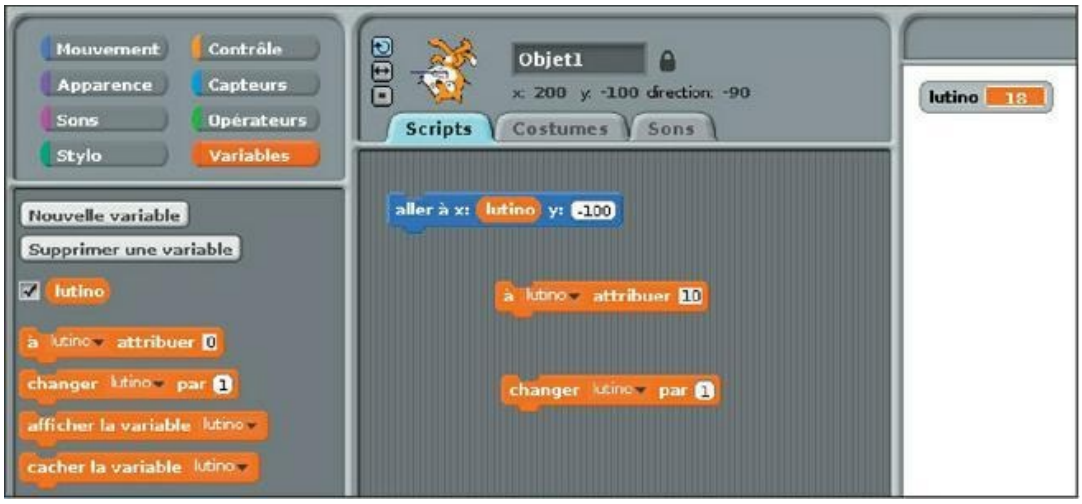


Figure 6.10 : Utilisation de blocs pour modifier une variable.

Les deux blocs `Mettre à` et `Ajouter` ont été déposés dans la zone des scripts, à l'écart, sans être associés à d'autres blocs dans le script. Pour modifier la valeur de la variable, il suffit de les utiliser tels quels. Je te conseille de positionner tes blocs de modification de valeur dans des endroits libres de la zone des scripts pour pouvoir plus facilement intervenir sur ces valeurs, puis les imbriquer dans un script une fois testés.

Les valeurs des variables dans les blocs

Nos précédents essais semblent fonctionner, mais le lutin n'a pas changé de place. Pourquoi ?

Je rappelle que les blocs de mouvement n'ont d'effet que lorsque tu cliques dedans ou lorsque tu demandes à Scratch d'exécuter le script qui les contient. Le fait de changer la valeur d'une variable utilisée dans un bloc n'a pas d'effet direct.

A priori, tu aurais pu penser le contraire. Pourtant, cela tient à de très bonnes raisons.

En effet, tu auras très souvent besoin de modifier une valeur sans provoquer une cascade d'actions. Scratch ne doit faire que ce que tu

veux qu'il fasse, et pas prendre des initiatives qui t'en feraient perdre le contrôle.

Pour que l'objet se déplace dans la scène, il faut utiliser la variable dans un bloc de mouvement puis cliquer le bloc ou lancer le script. Un point c'est tout.

La logique est la même pour les autres types de blocs. Ils n'utilisent la valeur de la variable que lorsqu'ils sont exécutés, par clic ou parce que Scratch atteint la ligne correspondante dans un script.

Montrer et cacher une variable

Devines-tu ce qui se passe lorsque tu cliques les blocs `Montrer la variable` et `Cacher la variable` ? Fais des essais.

Il n'y a rien de surprenant : le bloc `Cacher la variable` enlève la variable de l'affichage et l'autre la fait réapparaître.

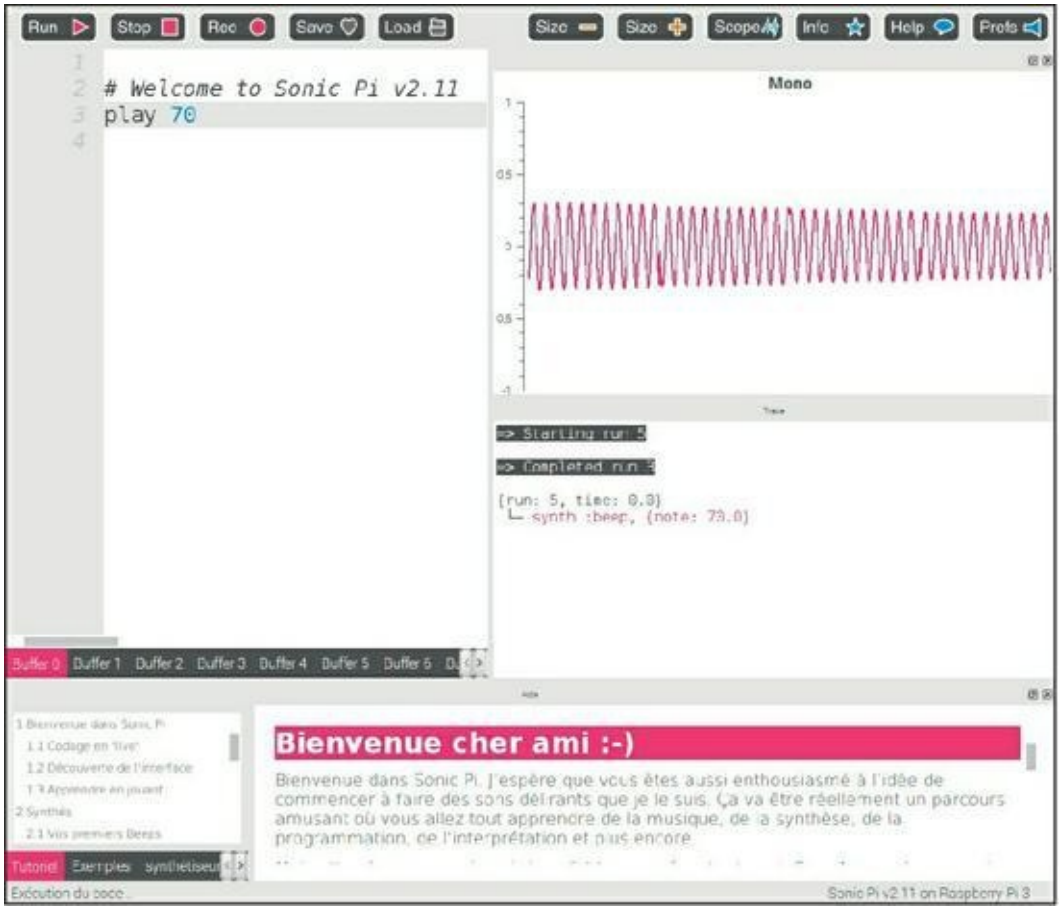
Ces deux options sont très utiles pour libérer l'espace d'affichage dans la scène une fois que le programme a été mis au point. En revanche, lorsque tu seras en pleine conception de ton projet, tu auras besoin de faire afficher certaines valeurs pour voir comment elles évoluent. C'est dans ce contexte que tu auras besoin de ces deux options.

Chapitre 7
Sonic Pi pour tes oreilles

Sonic Pi est un logiciel gratuit combinant un synthétiseur et un séquenceur. Tu n'as pas besoin de devenir un maître du clavier ou d'un autre instrument pour en jouer, il suffit d'écrire des instructions, comme dans un programme informatique.

Si tu n'y connais rien en musique, tu vas rapidement voir que Sonic Pi permet de créer toutes sortes de sons extraordinaires. Et si tu t'y connais en musique, tu pourras laisser libre cours à ton imagination !

À la différence du langage Python que nous allons découvrir plus loin et qui permet de faire toutes sortes de projets, le langage de Sonic Pi ne peut servir que dans Sonic Pi. Tu ne peux pas t'en servir pour faire des dessins ou pour créer un jeu vidéo ou un site Web. Cependant, certaines des instructions que tu vas découvrir dans Sonic Pi portent le même nom dans d'autres langages informatiques. La façon de les présenter et de les utiliser est cependant différente.



Pour démarrer Sonic Pi

Lorsque tu es devant le bureau graphique de ton Raspi, il suffit d'ouvrir le menu principal en haut à gauche puis d'accéder au sous-menu `Programmation`. Tu dois alors trouver une entrée pour `Sonic Pi` (Figure 7.1).

Figure 7.1 : La commande pour démarrer Sonic Pi.

Il te faut un peu de patience, car Sonic Pi nécessite quelques secondes pour démarrer. En effet, il doit charger un certain nombre d'ondes sonores. Tu sais que le chargement est terminé lorsque tu vois apparaître la fenêtre d'accueil par-dessus la fenêtre principale. Tu peux immédiatement refermer cette fenêtre en utilisant le bouton de fermeture dans l'angle supérieur droit de la petite fenêtre (pas de la grande).

De façon générale, tu pourras toujours refermer une sous-fenêtre en cliquant à nouveau le bouton qui a permis de l'ouvrir.

Configurer la sortie sonore

La sortie audio du Raspi fonctionne normalement sans problème. En revanche, il va te falloir ouvrir les paramètres de Sonic Pi, qui s'appellent des Préférences, selon la sortie choisie et le type de matériel audio que tu as branché.

Tu vois en haut une barre avec un certain nombre de boutons, le premier portant la légende Run à gauche. Si la fenêtre de Sonic Pi

n’a pas assez d’espace pour montrer tous les boutons, tu dois voir apparaître une petite flèche sur le bord droit de cette barre. Dans ce cas, clique cette flèche pour faire défiler les autres boutons afin de pouvoir accéder au dernier bouton de la série, le bouton des préférences.

Clique ce bouton `Préférences` pour accéder au panneau correspondant.

La Figure 7.2 montre les préférences du logiciel. Tu remarques notamment le curseur de réglage vertical qui permet de régler le volume sonore.

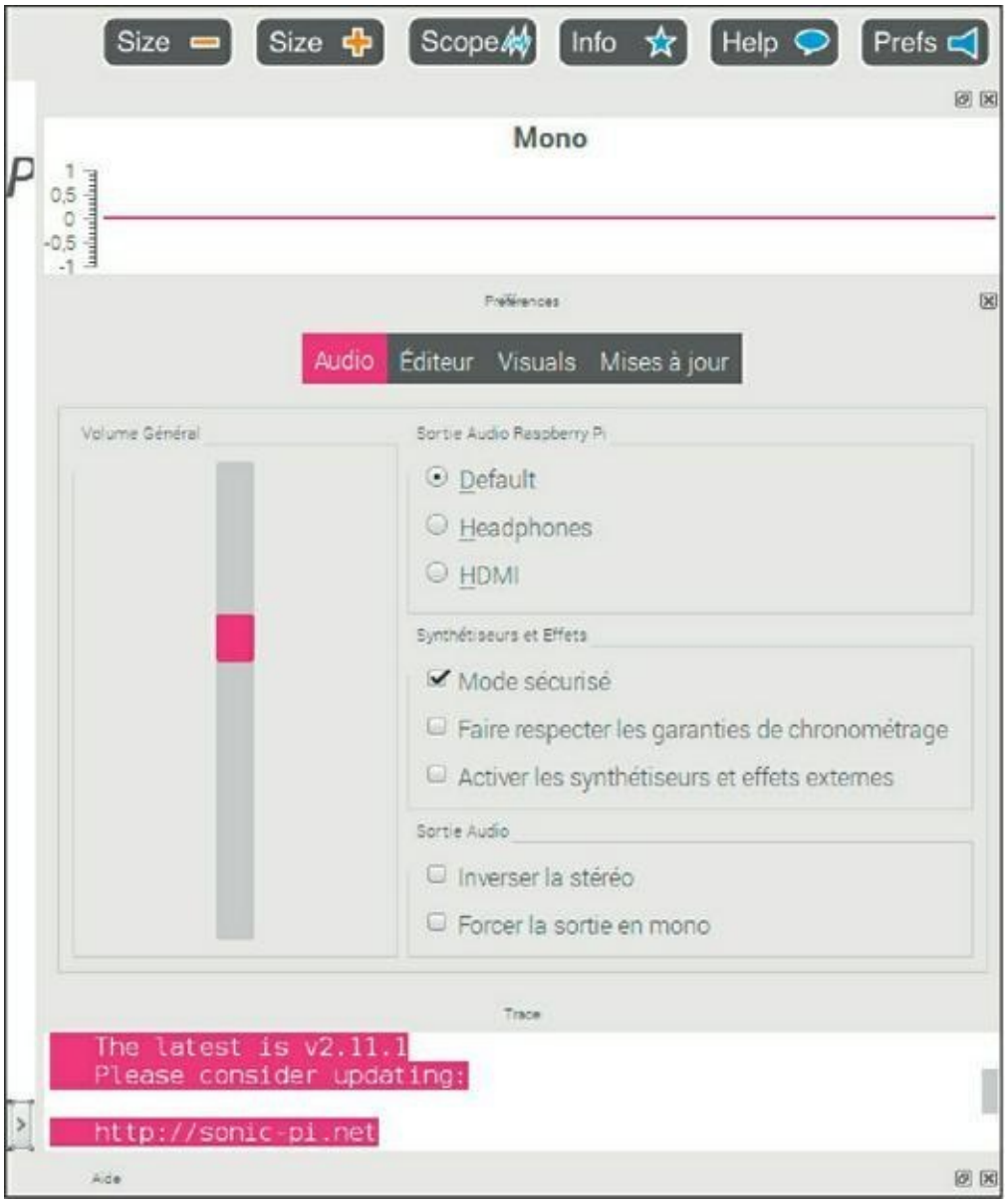


Figure 7.2 : Le panneau des préférences de Sonic Pi.

La sortie casque ou auxiliaire du Raspi n'est pas de très haute qualité. De plus, la puissance limitée du microcontrôleur ne permet pas d'envisager des créations de sons très complexes. Cela dit, tu vas pouvoir t'amuser suffisamment avec les synthétiseurs, les échantillons et les effets sonores, même si tu n'y connais rien en musique.

Les Raspberry Pi 3 sont plus puissants et offrent une qualité sonore bien meilleure que le circuit original. Il existe même des cartes

d’extension pour offrir une qualité sonore encore meilleure. Cherche sur le Web « sound card Raspberry » .

Si tu n’entends aucun son, c’est qu’il faut sans doute changer la sortie dans les préférences.

Tu peux par exemple avoir branché l’affichage sur un écran plat qui ne possède pas de haut-parleur. Dans ce cas, tu vas utiliser un casque sur la prise audio jack.

Si tu veux entendre le son sur les haut-parleurs de l’écran d’affichage, choisis l’option HDMI.

Si tu ne touches pas à ces options, le circuit va essayer de détecter la bonne sortie. Il est possible qu’il détecte un écran HDMI, mais si cet écran n’a pas de haut-parleur, il faudra manuellement venir changer les préférences dans ce panneau.

Pour refermer le panneau, utilise la petite case de fermeture en haut à droite ou clique à nouveau dans le bouton des préférences.

La solution la plus pratique consiste à brancher une enceinte avec une entrée auxiliaire à la sortie jack de ton circuit. Pense à baisser le volume avant de mettre sous-tension. On ne sait jamais quel sera le résultat avant d’avoir fait des essais.

Une première mélodie

Avant de faire une petite visite guidée du logiciel, essayons immédiatement de créer une petite mélodie, comme le montre la Figure 7.3.

Voici comment créer ta première mélodie :

1. Si nécessaire, démarre l’application Sonic Pi.

   Au démarrage, tu vois dans la fenêtre de code une seule commande :

```
play 70
```

2. Clique le bouton Run en haut à gauche.

   As-tu entendu un son ?

3. Si tu n'as rien entendu, ouvre le panneau des préférences et modifie les options en essayant chaque fois par un clic de Run.

4. Si tu as entendu un son, clique à nouveau dans la fenêtre de code et utilise la touche Retour Arrière afin de supprimer la valeur `70`.

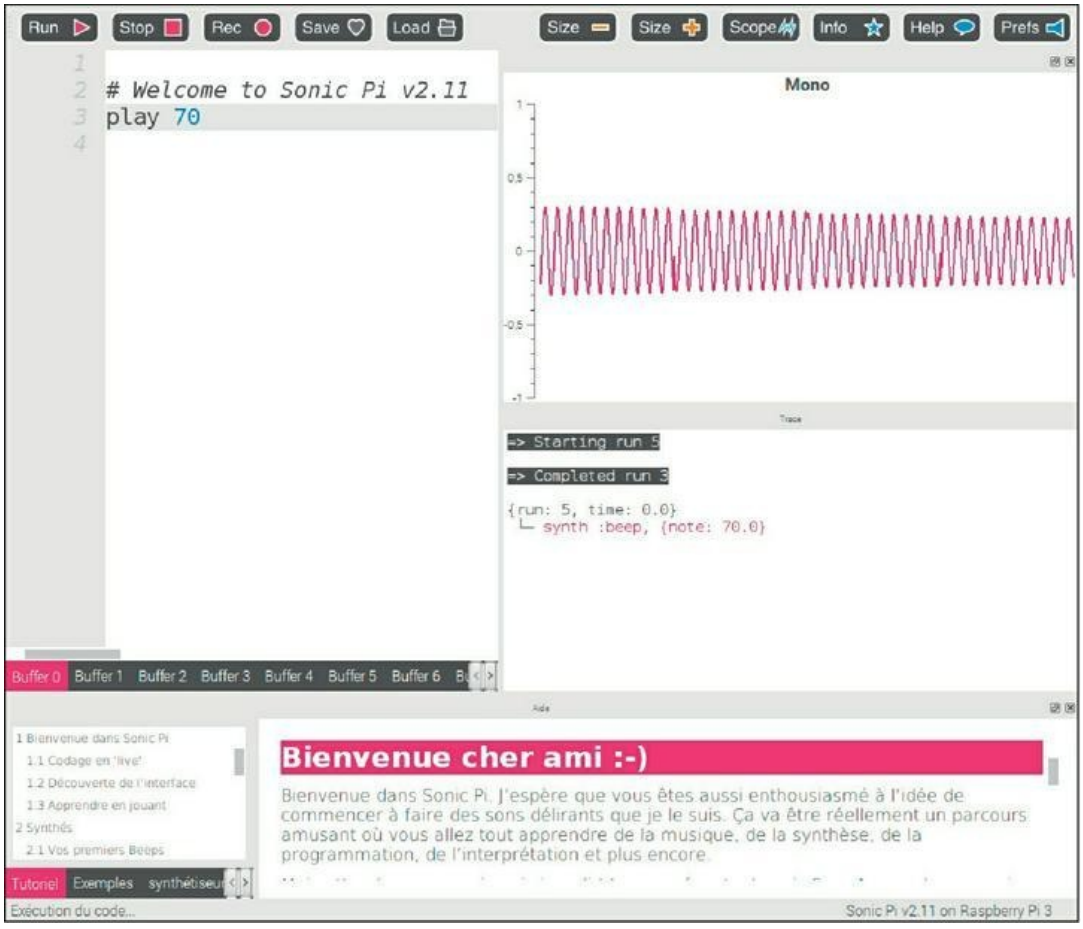


Figure 7.3 : Exemple de création d'une mélodie.

5. Saisis à la place la valeur `60`, en laissant bien une espace avant :

```
play 60
```

6. Relance le programme avec Run.

   Tu dois constater que le son est devenu plus grave.

   Tu peux bien sûr enchaîner plusieurs notes.

7. Modifie le code pour qu'il comporte les trois commandes suivantes et teste-le avec Run.

```
play 60
play 64
play 67
```

   Voici un petit embryon de mélodie.

8. Modifie le code pour qu'il se lise comme ci-après et démarre le morceau avec Run :

```
play 60
play 61
play 62
play 63
play 64
play 65
```

C'est moins esthétique, n'est-ce pas ?

Les musiciens savent quelles notes vont bien ensemble et lesquelles ont un effet moins intéressant. Cela dit, il faut savoir faire des exceptions pour maintenir l'intérêt. La plupart des morceaux

mélangent des notes harmonieuses et quelques notes moins harmonieuses.

La musique, c'est de la dentelle de silence

Tu n'es pas forcé d'enchaîner les notes comme nous venons de le faire. Sonic Pi permet de faire des pauses entre deux notes, comme ceci :

```
play 60
sleep 1
play 64
sleep 1
play 67
```

Lorsque la valeur de la commande `sleep` est égale à un, cela correspond à une mesure d'un temps. En musique, c'est la durée de base qui est divisée en moitiés, en quarts et en huitièmes. On appelle cela des croches, des doubles croches et des triples croches. Les musiques très rapides utilisent même des quadruples croches.

Sonic Pi a été conçu par un programmeur plutôt que par un musicien. Pour définir une durée, tu dois indiquer une fraction. Le Tableau 7.1 présente les différentes durées élémentaires.

Tableau 7.1 : Durées des sons et des silences.

Durée	*Silence*
Noire	Un temps
Croche	0.5
Double croche	0.25

Triple croche	0.125
Quadruple croche	0.0625

Saisis et lance le programme suivant :

```
play 60
sleep 0.125
play 64
sleep 0.5
play 67
sleep 0.25
play 64
sleep 0.125
play 60
```

En faisant varier les durées, tu crées un rythme, ce qui rend la série de notes beaucoup plus intéressante à entendre.

Le résultat est plus habituel lorsque le total des durées donne une valeur égale à un par exemple. Lorsque les durées ne tombent pas juste, tu obtiendras un résultat étonnant si tu fais jouer deux mélodies de durées différentes en même temps. Il se produira un décalage.

La composition en direct live

Sonic Pi a été inventé pour permettre de composer en direct, c'est-à-dire d'ajouter des commandes pour émettre des sons sans arrêter la musique en cours d'exécution.

On appelle cela du *live coding*.

Une visite guidée

Découvrons maintenant le contenu de la fenêtre principale de Sonic Pi.

La fenêtre d’écriture du code

C’est dans la fenêtre de code que tu écris les commandes qui vont générer de la musique. C’est un éditeur de texte spécifique pour Sonic Pi.

Observe la partie inférieure de cette fenêtre : tu vois des petits rectangles avec la mention `Buffer`. Tu peux cliquer chacun des boutons pour accéder à

10 fenêtres de code différentes.

Est-ce que cela signifie que tu peux faire jouer plusieurs scripts en même temps ? Absolument ! C’est grâce à cette possibilité que tu peux vraiment faire de la composition en direct. Tu peux lancer le contenu d’une fenêtre de code tout en écrivant le contenu d’une autre, puis lancer la restitution de cet autre contenu qui viendra s’ajouter au premier. Chaque fenêtre peut être contrôlée, c’est-à-dire démarrée et arrêtée, indépendamment.

Si tu fais une erreur dans l’écriture du code, Sonic Pi le détecte et fait apparaître une fenêtre de message d’erreur sous la fenêtre de code. Les messages sont en anglais et ne sont pas toujours faciles à comprendre, mais cela devrait te donner suffisamment d’indices pour savoir ce qui cloche. La Figure 7.4 donne un exemple de message d’erreur.

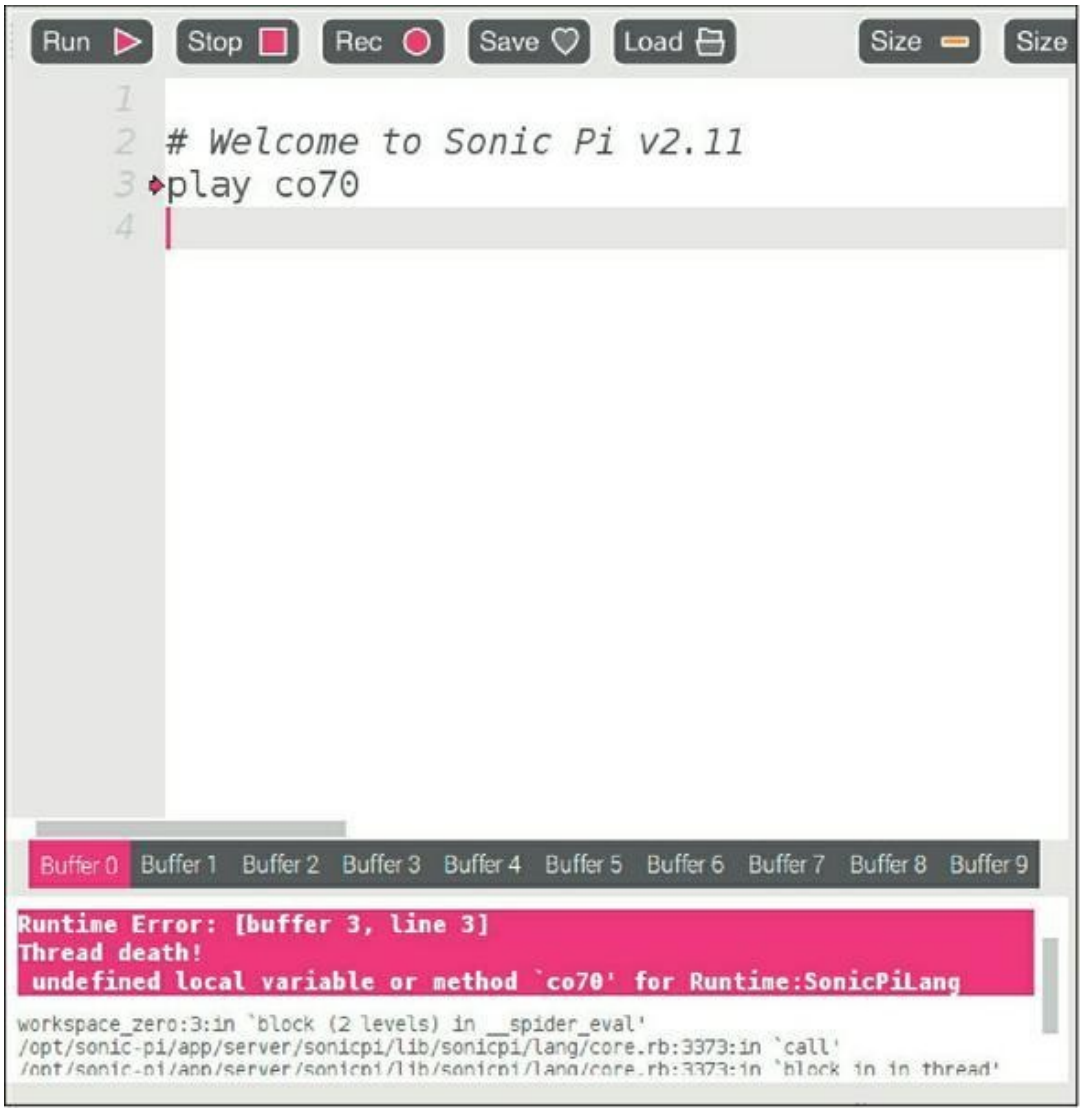


Figure 7.4 : Exemple de message d'erreur.

La fenêtre de suivi d'exécution Trace

Sur la droite se trouve une fenêtre qui affiche des messages en cours d'exécution. Dès qu'une note est émise, un message apparaît. Quand tu en sauras plus sur Sonic Pi, tu pourras même faire apparaître des messages personnels dans cette zone.

Lors de tes débuts, tu peux ne pas tenir compte des messages qui apparaissent ici. Ils deviendront de plus en plus clairs au fur et à mesure de ta progression.

Les fenêtres d'aide et d'apprentissage

Tout en bas à gauche se trouve la fenêtre d'aide. Elle comporte deux panneaux.

Le petit panneau à gauche contient la table des matières de la rubrique sélectionnée.

Il suffit de cliquer le bouton d'une rubrique puis l'intitulé d'un article pour en faire apparaître les détails dans le panneau de droite.

Voici les rubriques disponibles :

- Tutoriel : il s'agit d'une longue série de leçons qui te permettent d'avancer par autoapprentissage. La traduction en français de ces leçons est en cours.
- Exemples : ce sont plusieurs projets dont tu peux t'inspirer.
- Synthétiseur : un réservoir de sons électroniques dans tous les timbres : basse, percussions, effets sonores, instruments classiques, etc.
- FX : une série de modules pour modifier les sons. C'est ici que tu trouveras par exemple un module pour ajouter de la distorsion, ce qui donne parfois un effet très intéressant.
- Échantillons : un réservoir de sons prédéfinis. Il se distingue de la rubrique Synthétiseur par le fait que ce

sont des sons échantillonnés sans traitement, ce qui convient particulièrement aux sons de batterie, aux boucles de rythmes, aux sons d'ambiance et autres sons difficiles à synthétiser.

» Langage : cette section très utile rappelle toutes les commandes et tous mots-clés utilisables dans tes scripts Sonic Pi.

Si la résolution de ton écran ne permet pas d'afficher les noms de toutes les rubriques, tu peux utiliser la petite flèche du côté droit pour voir celles qui sont cachées. Pour plus de confort de lecture, utilise le séparateur de panneaux horizontal en amenant doucement le pointeur au-dessus de la fenêtre d'aide ; fais-le glisser vers le haut pour augmenter la hauteur de l'affichage. Bien sûr, cela réduit temporairement la surface de la fenêtre de code.

La Figure 7.5 montre la fenêtre de tutoriel agrandie.

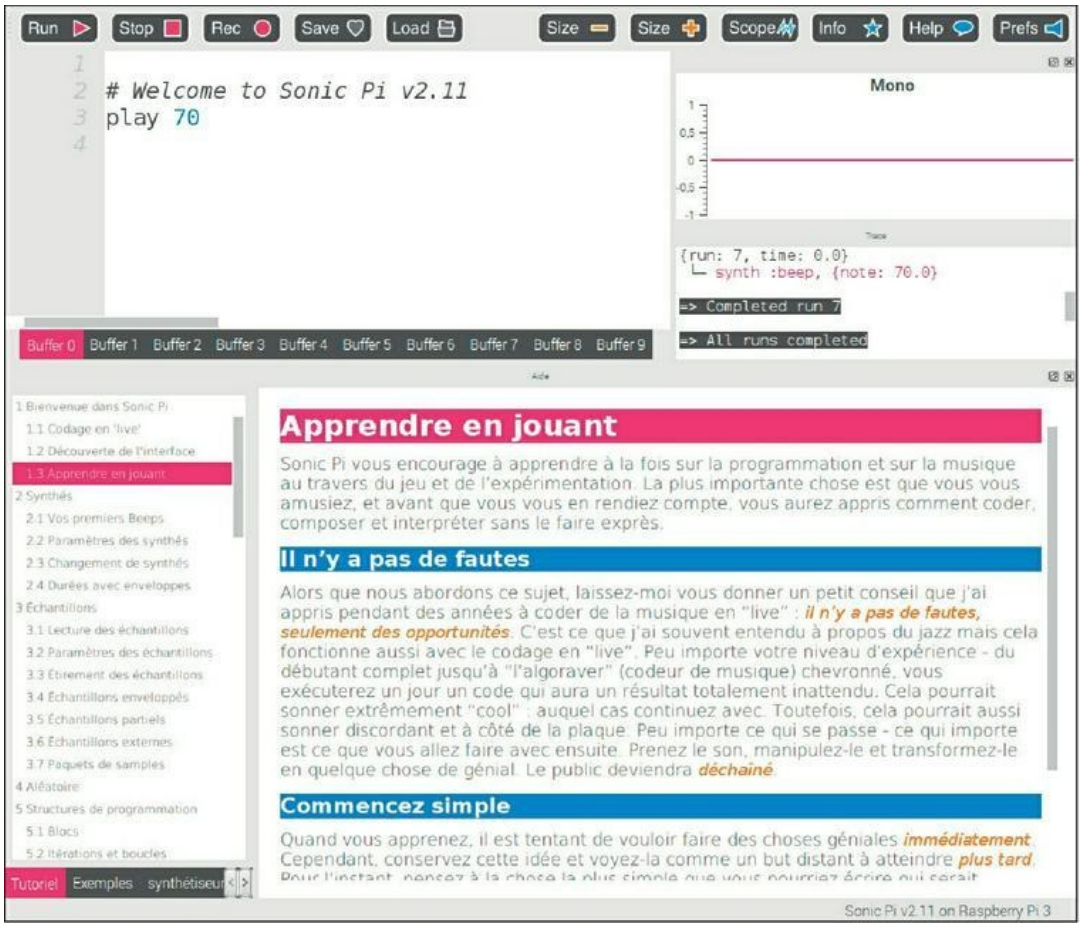


Figure 7.5 : Affichage en grand format de la fenêtre du tutoriel.

Tous les exemples du tutoriel peuvent être directement réutilisés. Il suffit de sélectionner le code dans la fenêtre du bas (rouge dans les tutoriels et bleu dans les exemples) puis de cliquer-droit pour choisir `Copier`. Clique ensuite dans un espace de code libre puis clique-droit pour choisir `Coller`. Il ne te reste plus qu'à lancer l'exécution avec `Run`. Pour nettoyer un espace de code, clique-droit puis choisis `Sélectionner tout` et frappe la touche Suppr.

La barre d'outils

Tout en haut de la fenêtre, tu disposes d'une série d'outils qui contrôlent les principales fonctions de l'application. La plupart de ces

boutons laissent deviner leur effet par simple lecture, mais ce n'est pas le cas de tous.

- `Run` : exécute le code de l'espace de code actuellement affiché.
- `Stop` : arrête l'exécution et donc l'émission de sons dans tous les espaces de travail.
- `Save` : enregistre dans un fichier le code de l'espace de code actif.
- `Rec` : enregistre l'exécution sonore dans un fichier son. Dès que tu cliques `Stop`, Sonic Pi te demande d'indiquer le nom du fichier dans lequel l'exécution doit être stockée.
- `Size+` et `Size -` : ces deux boutons servent à augmenter et réduire la taille des caractères dans les fenêtres de code. Cela n'a aucun effet sur le sens.
- `Info` : affiche la fenêtre des informations au sujet de l'application. Tu n'en auras besoin que pour vérifier le numéro de version.
- `Help (Aide)` : affiche et masque le panneau d'aide.
- `Prefs` : sert à accéder aux préférences audio du Raspi comme déjà décrit en début de chapitre. Si ce bouton n'est pas visible, tu disposes d'une petite flèche à droite de la barre d'outils pour le faire apparaître.

Il peut arriver que tu fasses disparaître la barre d'outils par mégarde. Pour faire réafficher les boutons, clique dans la barre située à droite

du mot `Trace` dans la fenêtre de Trace puis choisis `Outils` dans le menu qui apparaît.

Aide à la saisie

Dès que tu saisis le début du nom d'une commande dans la fenêtre de code, Sonic Pi te propose la fin du nom. Il se base sur tous les mots qui commencent de la même manière que ce que tu as saisi.

Tu peux choisir l'une des propositions en utilisant la souris et la barre de défilement ou bien tu peux continuer à saisir pour diminuer le nombre de choix proposés. Quand le menu montre la commande que tu veux saisir, il suffit de valider par Entrée. Le menu de suggestion est visible dans la figure suivante.

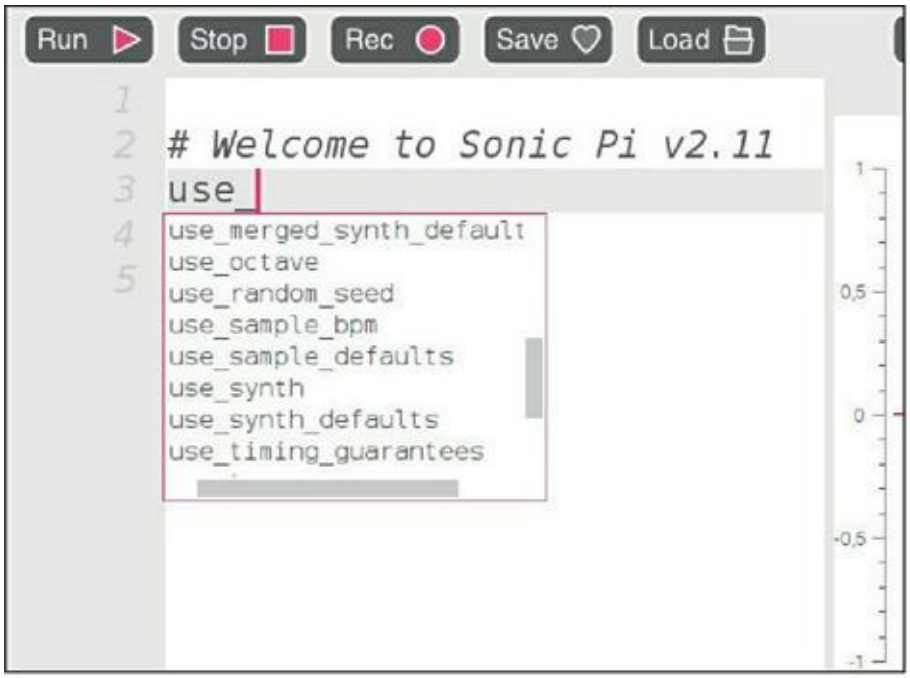


Figure 7.6 : Exemple d'aide à la rédaction d'une commande.

Cette fonction très répandue de nos jours porte le nom d'aide à la rédaction ou *code completion*. La plupart des éditeurs de programmation offrent cette possibilité qui peut faire gagner beaucoup de temps et éviter toutes sortes de fautes de saisie. N'hésite donc pas à t'habituer à t'en servir.

Désigner une note par un numéro

J’ai dit qu’il n’était pas nécessaire d’être musicien pour savoir produire des sons avec Sonic Pi, mais il reste utile de savoir comment faire.

Les notes de musique sont des hauteurs ou des fréquences. Sonic Pi permet de désigner les notes de deux façons différentes.

Tu peux indiquer les notes avec un numéro entre 0 et 127. Les valeurs inférieures à 24 sont pratiquement inaudibles, car trop graves. De même, les notes supérieures à 100 sont tellement aiguës qu’elles peuvent faire mal aux oreilles.

Les notes les plus utilisées en musique sont celles situées entre les valeurs 40 et 70, mais tu peux sortir de ces limites comme bon te semble.

Lorsque tu désignes une note par numéro, tu peux indiquer des valeurs non entières, comme ceci :

```
play 59.95
play 60
play 60.05
```

Tu remarques qu’il faut utiliser un point et pas une virgule. Lorsque tu fais jouer des notes dont les valeurs (les fréquences) sont très proches comme dans l’exemple précédent, tu produis un son plus épais.

Si tu augmentes encore les écarts entre les notes, tu produis des dissonances.

Lorsque les notes sont un peu plus écartées que dans l’exemple, cela devient de la musique microtonale qui est réservée aux compositeurs de musique contemporaine. Ces sons pourront te sembler très bizarres.

Désigner une note par un nom

Si tu t'y connais un peu en musique, tu sais peut-être que les Anglo-Saxons utilisent les lettres de A à G pour indiquer les notes. Dans Sonic Pi, tu peux utiliser ces lettres en les faisant précéder d'un signe deux-points :

```
play:e
```

Si tu ajoutes une valeur entre 0 et 10 après la lettre, tu définis l'octave, c'est-à-dire le nombre de multiples de la note la plus basse, celle à l'octave zéro :

```
play:e2
play:e5
```

Une note qui est située à une octave au-dessus de la précédente possède une fréquence double. Elle porte le même nom de note, mais elle est plus aiguë.

Tu peux enfin altérer une note en l'augmentant un peu avec la lettre `s` pour obtenir un dièse (le S est l'initiale de « sharp ») ou descendre d'un demi-ton une note en utilisant le `b` qui veut dire bémol :

```
play:c
play:cs
play:cb
```

Découvrir les synthétiseurs

Pour faire de la musique avec Sonic Pi, tu décides d'abord quelle note il faut jouer puis tu choisis le son à utiliser, qu'il s'agisse d'un synthétiseur ou d'un échantillon.

Pour choisir d'utiliser un synthétiseur, tu utilises cette commande :

```
use_synth :nom_synthe
```

Pour obtenir la liste de tous les noms de synthétiseurs disponibles, il suffit d'ouvrir la page `Synthétiseur` dans le coin inférieur gauche. N'hésite pas à essayer tous les synthétiseurs pour trouver le son que tu préfères.

Fais bien attention au fait que le signe deux-points doit être collé au début du nom du synthétiseur, sans espace. Il ne doit pas être accroché à la fin du mot `use_synth`. Nombreux sont les débutants qui font cette erreur.

Voici par exemple comment jouer quelques sons en changeant de timbre à chaque fois :

```
use_synth :fm
play 60
sleep 0.25
use_synth :mod_beep
play 60
sleep 0.25
use_synth :growl
play 60
sleep 0.25
use_synth :hollow
play 60
```

Les paramètres des synthétiseurs

À la différence des échantillons, les synthétiseurs sont modulables grâce à des paramètres. Toute une série de paramètres se retrouve dans tous les synthétiseurs, mais d'autres sont spécifiques.

Voici comment modifier un paramètre en l'ajoutant après la hauteur de note :

```
play 60, amp: 0.1
```

Ce paramètre, qui porte le nom `amp`, permet de régler le volume sonore.

Pour indiquer plusieurs paramètres, il suffit de les séparer par des virgules :

```
play 60, amp: 0.1, pan: -1
```

Les paramètres utilisables dans la plupart des synthétiseurs sont réunis dans le Tableau 7.2. Les trois paramètres portant le nom Attack, Decay et Release doivent être suivis d'une valeur indiquant le nombre de secondes de l'effet à partir de zéro. La durée maximale n'est pas limitée.

Tableau 7.2 : Paramètres principaux des synthétiseurs.

Nom	*Effet et plage de valeurs*
amp	Règle le volume entre zéro et un.
pan	Déplace le son entre les deux haut-parleurs stéréo, de -1 à +1.
attack	Contrôle la vitesse d'arrivée du son.
decay	Contrôle la vitesse d'atteinte du volume de maintien du son.
sustain	Contrôle la durée de maintien du son entre zéro et un.
release	Contrôle la vitesse de disparition du son.

Rien ne t’empêche de faire en sorte qu’un son mette des heures à s’établir puis à s’évanouir, mais ce n’est pas très intéressant à écouter.

Les paramètres par défaut

Chaque son de synthétiseur dispose de valeurs prédéfinies pour tous ses paramètres. Lorsque tu demandes à jouer un son avec un synthétiseur, ce sont les valeurs par défaut qui sont utilisées, sauf si tu indiques une valeur spéciale pour un paramètre.

Pour afficher la liste des valeurs par défaut des paramètres, il suffit d’accéder à la page `Synthétiseur` dans la fenêtre d’aide en bas. Sous la description du son, tu dois trouver une série de boîtes qui indiquent les valeurs pour tous les paramètres reconnus par ce synthétiseur. La valeur par défaut est indiquée.

Si tu regardes par exemple les détails du synthétiseur qui porte le nom `dsaw`, tu pourras vérifier que la valeur par défaut pour `Sustain` est égale à zéro, et que celle pour `Cutoff` est égale à 100. Observe le contenu la Figure 7.7.

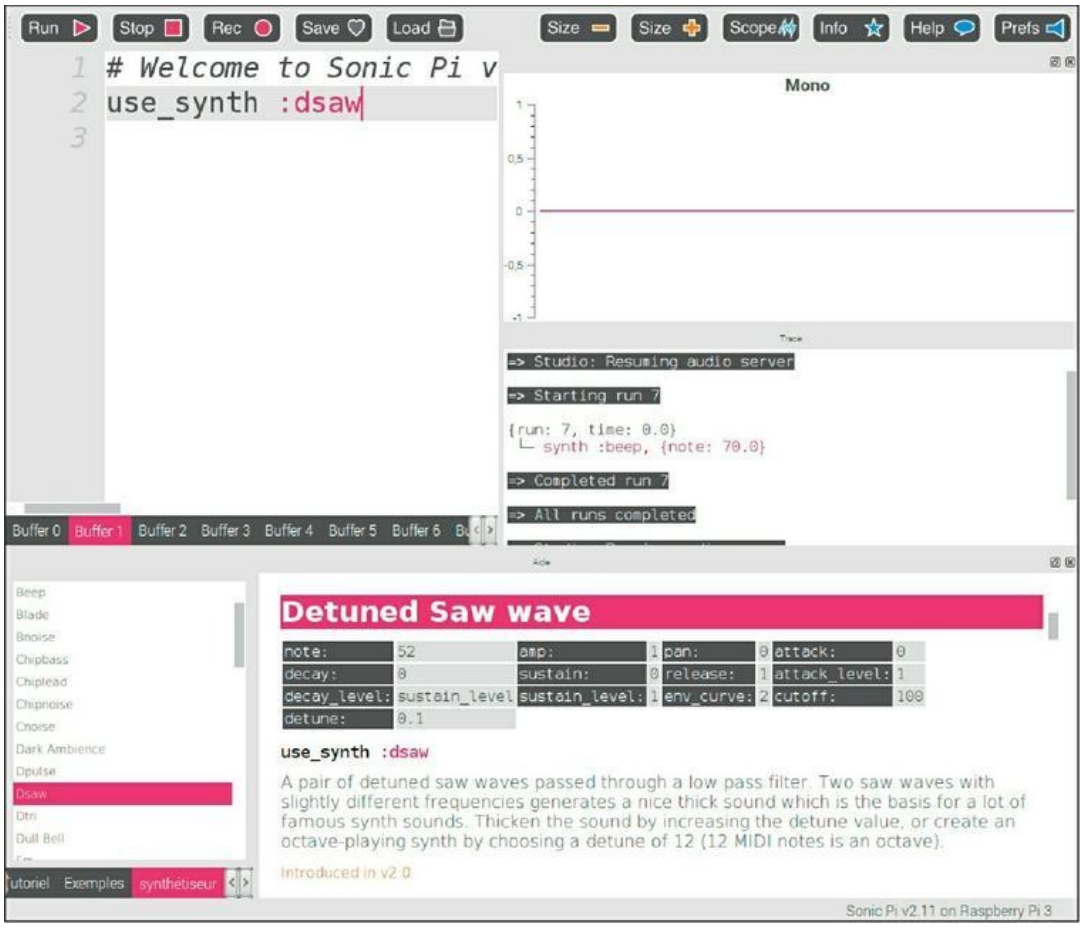


Figure 7.7 : Page de description du synthétiseur dsaw.

Ce n’est pas grave si tu ne sais pas à quoi sert chacun des paramètres. Tu ne risques pas de casser quoi que ce soit. Poursuis tes essais. Les noms des paramètres sont en général les mêmes que ceux utilisés habituellement dans le monde des synthétiseurs et dans la musique électronique. N’hésite pas à chercher le nom du paramètre sur Internet pour en savoir plus.

Pour aller plus loin

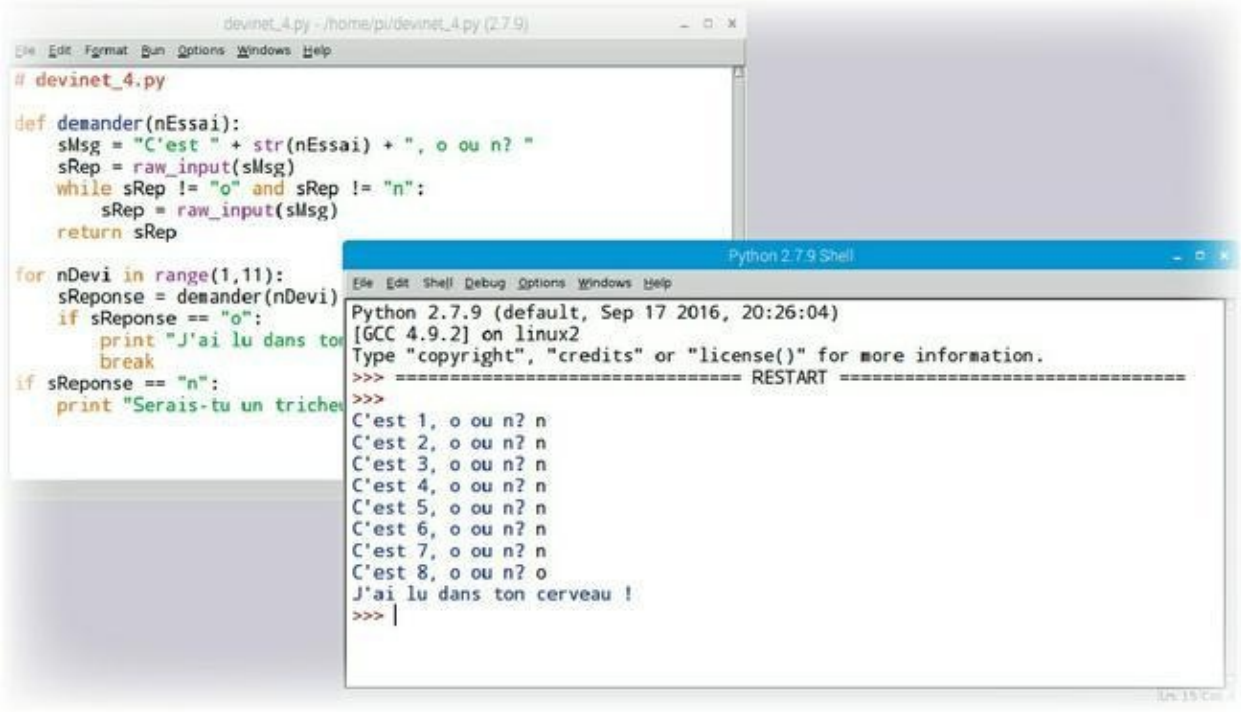
Tu en sais presque assez pour commencer à écrire quelques mélodies. N’hésite pas à profiter du fait que tu peux changer de synthétiseur et jouer sur les paramètres en plein milieu de tes morceaux, même avec une seule note.

Si tu t’y connais un peu en musique, tu peux repiquer les noms des notes d’une chanson que tu aimes pour la recomposer dans Sonic Pi, et même créer des accords.

Bien sûr, puisque Sonic Pi utilise des commandes, tu peux très facilement faire répéter des sons ou des mélodies en faisant des copier/coller de tout un bloc de commandes. Surtout, n’hésite pas à plonger dans le tutoriel qui est disponible en français pour les premières leçons, et qui contient une foule d’exemples qu’il suffit de copier/coller dans une fenêtre de code pour entendre immédiatement le résultat. Sache qu’avec Sonic Pi, tu ne risques jamais de faire une erreur, seulement des résultats d’exploration.

Si le fait d’utiliser cette application te donne envie d’en apprendre plus sur la théorie musicale, puise dans les ressources sur Internet. La musique est une véritable science qui peut te tenir en haleine toute ta vie. En quelques soirées, tu peux apprendre l’essentiel pour devenir autonome.

Plus loin en programmation



Au programme de cette partie :

Chapitre 8
Découvrons Python

Les chercheurs en informatique adorent créer sans cesse de nouveaux langages de programmation (ou infolangages). De nos jours, il en existe plusieurs centaines, mais une poignée seulement sont utilisés à vaste échelle.

Dans ce chapitre, nous allons découvrir l'un des infolangages les plus utilisés. Il porte un nom de reptile : Python. C'est un des langages les plus faciles à apprendre à utiliser, mais c'est malgré tout un langage complet, utilisé dans des applications industrielles et scientifiques. En choisissant d'apprendre le langage Python, tu te prépares à devenir capable de créer de vrais programmes, et même pourquoi pas, de les commercialiser.

Premier contact avec Python

Nous avons découvert l'atelier Scratch, ce qui t'a permis de te faire une première idée de la façon dont on peut contrôler le comportement d'un ordinateur. Tu peux déjà faire un certain nombre de choses avec Scratch, mais son domaine d'emploi reste tout de même limité. Tu ne peux par exemple pas créer avec Scratch une application graphique complète.

Dès que tu as besoin de créer un véritable projet, il faut pouvoir contrôler plus précisément l'ordinateur. La façon la plus répandue de nos jours consiste à écrire les unes après les autres des lignes d'instructions qui obéissent aux règles d'un langage. Une fois que ces instructions sont écrites, un outil spécial appelé *compilateur* les analyse et, s'il ne trouve aucune erreur, il produit du code binaire qui pourra être exécuté par le microprocesseur de ton Raspberry Pi.

La puissance des infolangages découle de la possibilité de créer des fonctions et d'utiliser des fonctions déjà écrites par d'autres. À partir du moment où un programmeur a écrit un petit programme pour afficher un pixel sur un écran en choisissant sa couleur, tous les autres programmeurs peuvent réutiliser ce petit programme (s'il le partage), ce qui leur évite de perdre du temps à écrire les mêmes instructions.

Les instructions que tu vas écrire constituent le code source du programme. Ce sont des mots-clés souvent inspirés de l'anglais. Pour un débutant francophone, cela ne pose aucun problème puisqu'au départ, un anglophone ne sait pas non plus ce que chaque mot-clé représente exactement.

Le langage Scratch est tout à fait particulier, puisqu'il utilise des blocs contenant des noms d'actions. Les véritables langages de programmation utilisent des noms de commandes, de variables et de fonctions, des directives et des valeurs en les délimitant par des espaces, et d'autres signes de ponctuation.

Pour créer le texte source d'un programme, tu vas utiliser un éditeur de texte. C'est une version simplifiée de traitement de texte. Certains outils d'édition sont dédiés à un langage et permettent donc de tester directement le programme, sans quitter l'éditeur.

Réveiller son Python préféré

Le système d'exploitation RaspBian qui anime ton Raspi offre deux versions du langage Python, la version 2.x et la version 3.x. Toutes deux sont disponibles, comme tu peux t'en douter, dans le sous-menu `Programmation` du menu principal (Figure 8.1).

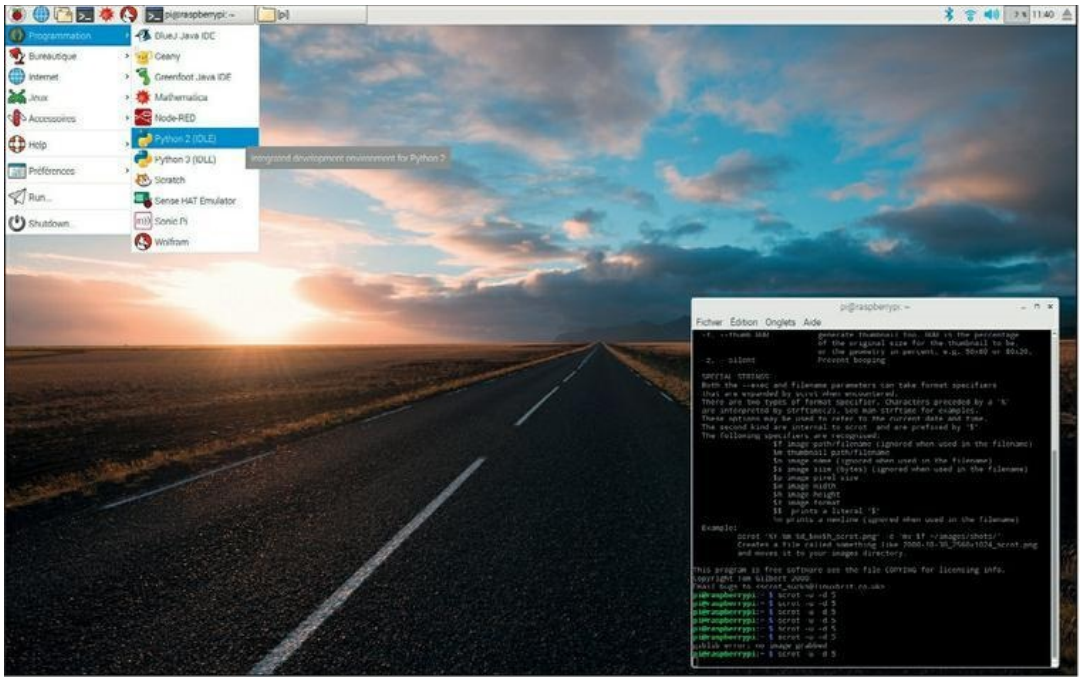


Figure 8.1 : Accès aux deux versions de Python.

S'il y a deux versions, tu pourrais te dire qu'il faut utiliser la plus récente, non ? Pas dans le cas de Python, car son histoire est particulière. La version 3 existe depuis maintenant 10 ans, mais il y a eu tellement de programmes et de librairies de fonctions conçus pour la version 2 que de nombreuses personnes continuent à utiliser cette version 2. En effet, certains programmes (pas tous) prévus pour la version 2 ne fonctionnent plus dans la version 3.

Dans ce livre, nous serons conservateurs et utiliserons seulement l'ancienne version 2. N'hésite cependant pas à découvrir plus tard la nouvelle version qui ouvre de nouveaux horizons.

Voici comment démarrer l'environnement de création de Python 2 :

1. Ouvre le menu principal représenté par une framboise.
2. Ouvre le sous-menu `Program mation` et sélectionne la commande `Python 2 (IDLE)`.

Au bout de quelques instants, tu vois apparaître une fenêtre quasiment vide. Si tu regardes bien le texte qui est apparu en haut, tu peux confirmer que c'est bien une sous-version de la version 2 de Python qui a été démarrée dans cette fenêtre.

La barre de titre de la fenêtre contient le mot anglais *shell* qui signifie « coquille » . Pourtant, ce programme n'a aucun rapport direct avec un quelconque fruit de mer. Il faut comprendre ce terme shell comme un environnement dans lequel tu peux utiliser un certain nombre de commandes pour agir sur ton ordinateur. Le shell comporte une invite qui est ici la série de trois signes >. On parle également d'interpréteur de commandes, car les commandes que tu saisis sont immédiatement prises en compte dès que tu valides par la touche Entrée.

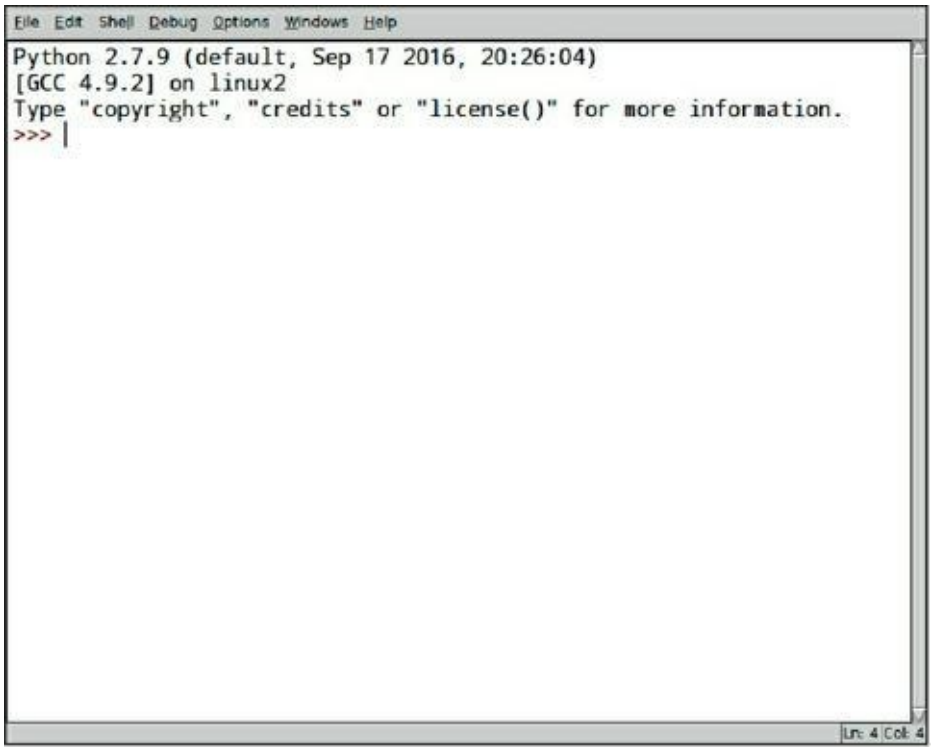


Figure 8.2 : La fenêtre d'exécution de Python 2 IDLE.

Tu as peut-être remarqué que la date affichée en haut de la fenêtre n'est pas la bonne. C'est la date de production de ta version de

Python. Une fois que tu le sais, tu es rassuré.

Configurer son Python

Avant de commencer à écrire du code source, il n'est pas inutile d'ajuster la taille du texte affiché dans les fenêtres de Python. Tu peux ignorer cette étape si le texte te paraît déjà à la bonne taille, par exemple si tu as un très grand écran pour l'affichage.

Pour ajuster la taille (le corps) du texte, procède ainsi :

1. Tu vois la barre de menu dans le haut de la fenêtre de Python shell ? Clique la rubrique `Options`.
2. Dans le menu, choisis `Configure IDLE` (Figure 8.3).

   Tu vois apparaître la fenêtre des paramètres, `Settings`.

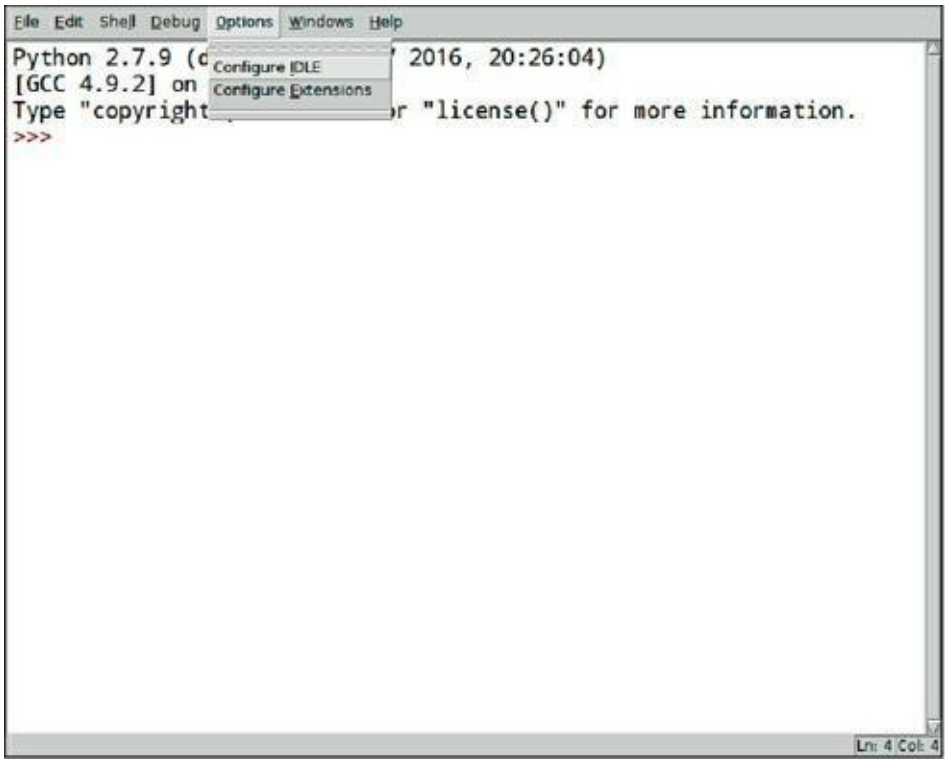


Figure 8.3 : Le menu d'accès à la configuration.

3. Dans la page des polices, `Font`, clique la zone de saisie `Size`.

4. Choisis une taille de police dans la liste déroulante (Figure 8.4).

   La taille qui est proposée au départ, c'est-à-dire la taille par défaut, est égale à 10. C'est souvent trop petit. Dans ce livre, j'ai opté pour une taille de 14, afin que les captures d'écran soient faciles à lire. Si c'est trop gros pour toi, choisis un corps 12.

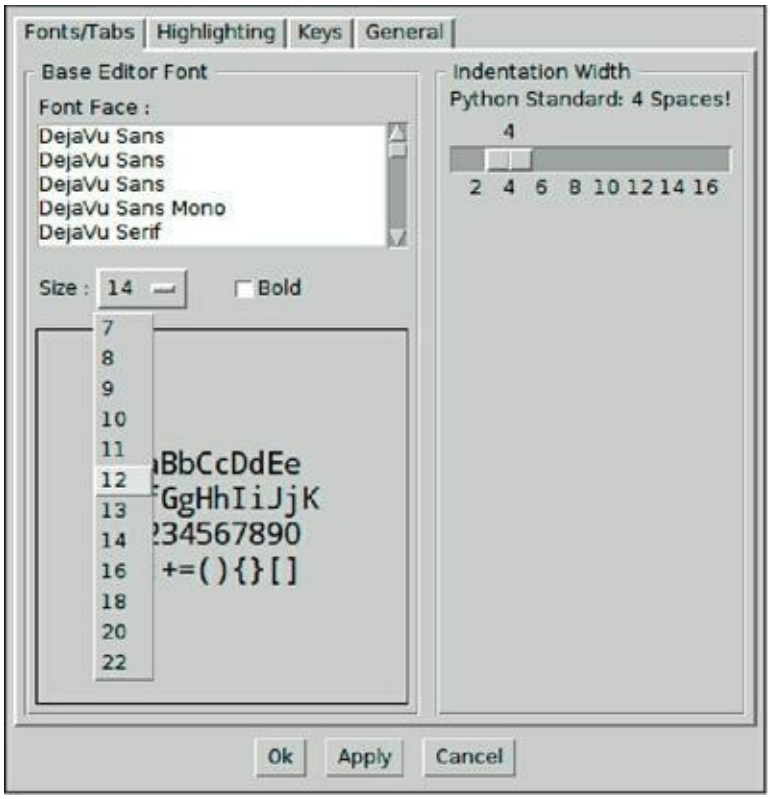


Figure 8.4 : Choix d'une taille de caractères.

5. Valide ton choix en cliquant OK.

Tu constates que la surface de la fenêtre est adaptée à la nouvelle taille du texte.

Si tu choisis un corps vraiment gros, il est possible que la fenêtre déborde de la surface de l'écran. Cela n'a pas d'effet négatif, mais cela peut devenir gênant pour lire la totalité du contenu.

Un Python mathématicien

Tu peux dès maintenant profiter de l'énorme puissance de calcul qu'offre le langage Python. Dans la fenêtre, la ligne de l'invite (les trois signes >) contient la barre verticale clignotante qui montre où se trouve le curseur. C'est là que va apparaître le prochain caractère frappé sur le clavier. L'invite « t'invite » à saisir une commande ou une équation. Le principe est le même que l'invite du système Linux lorsque la machine démarre en mode texte ou lorsque tu ouvres une fenêtre de terminal. Nous y reviendrons plus tard. Pour l'instant, testons notre nouveau langage.

Saisis la petite formule suivante, puis valide ta saisie en frappant la touche Entrée :

```
1 + 1
```

Incroyable ! Python résout ce problème presque instantanément (Figure 8.5).

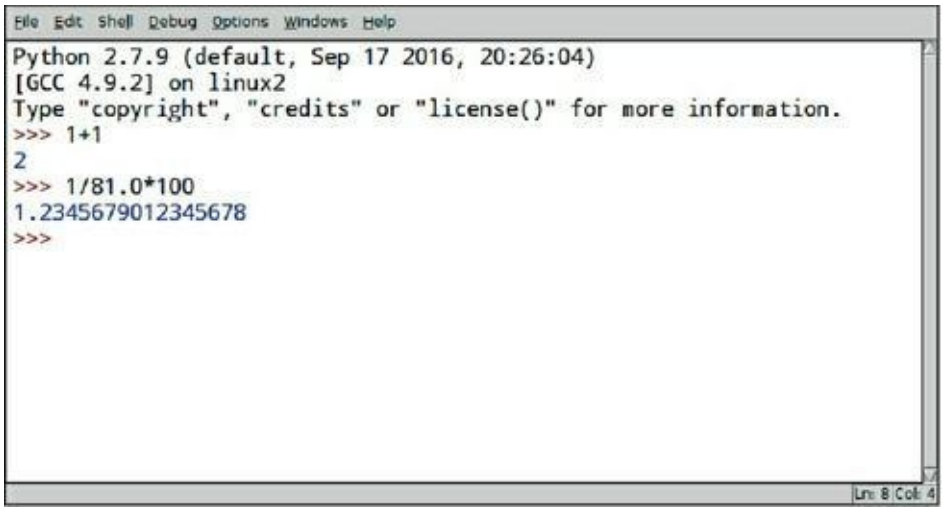


Figure 8.5 : Premier calcul dans Python.

Bien sûr, Python sait faire des calculs bien plus complexes, comme celui-ci :

```
1 / 81.0 * 100
```

N'oublie pas de frapper la touche Entrée pour valider. Tu constates que Python est très précis dans son résultat.

Python est plus précis qu'une calculatrice de bureau puisqu'il offre 16 chiffres après la virgule. Fais un essai avec une autre calculatrice pour voir si le résultat est le même. Cette meilleure précision pourra être très utile dans tes calculs, même au niveau scolaire. Python est très apprécié des scientifiques.

Ne pas avoir peur de faire des erreurs !

La figure 8.6 montre ce qui se passe lorsque l'interpréteur de Python ne comprend pas ce que tu lui demandes de faire. C'est en général parce que tu as fait une faute de frappe ou que tu as saisi n'importe quoi, comme je l'ai fait ici. Le message est en anglais ; il signifie tout simplement que ce qui a été saisi est incompréhensible pour l'interpréteur du langage.

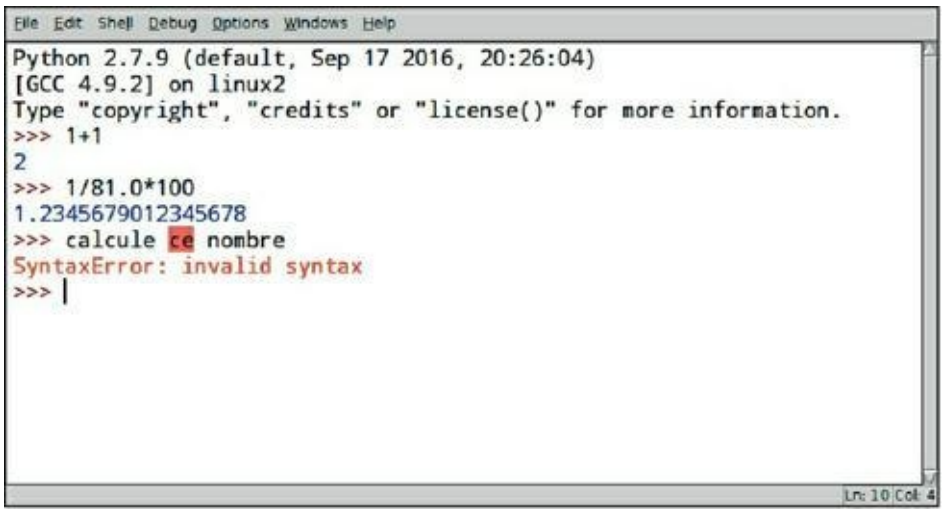


Figure 8.6 : Exemple de message d'erreur dans Python.

Python sait afficher une large gamme de messages. Plus tu vas t'exercer à programmer en Python, plus les messages te deviendront compréhensibles, même s'ils sont en anglais. Il en est ainsi dans tous les langages de

programmation. On aurait pu espérer que les messages soient plus explicites, mais en quelques heures, tu sauras déjà comment réagir aux principaux messages d'erreur.

Des données en mémoire

Les deux piliers d'un langage de programmation informatique sont le code et les données. Les données sont les valeurs qui sont stockées en mémoire. Dans Python, tu prépares l'utilisation d'une donnée en lui attribuant un nom, ce qui va automatiquement lui réserver un emplacement dans la mémoire.

Chaque emplacement mémoire correspond à une donnée. Une fois que la donnée est définie, tu peux la relire et la modifier.

Commençons par créer notre première variable en choisissant le contenu le plus simple qui soit : une valeur numérique entière. Nous verrons plus tard comment stocker des mots et des phrases.

En langage informatique, chaque emplacement mémoire est une variable, ce qui te laisse deviner que le contenu doit pouvoir changer.

Quand tu en sauras plus sur Python, tu découvriras un genre de variable plus complexe : le tableau (*array*). C'est en fait un groupe de variables, un emplacement qui en contient plusieurs. Imagine par exemple un tableau contenant 30 variables pour les 30 moyennes générales des élèves d'une classe. Les tableaux permettent d'éviter de devoir jongler avec des dizaines de noms de variables. Tu découvriras ce type plus tard.

Créer sa variable

Pour créer une variable et y stocker une valeur, saisis la ligne suivante, sans oublier de valider la saisie par la touche Entrée :

```
ma_varinum = 1
```

Cette instruction demande de réserver un espace suffisant en mémoire pour y stocker un nombre entier. Le nom va permettre, dans la suite des opérations, d'accéder à la valeur.

Tu constates que je propose d'utiliser un caractère de soulignement dans le nom de la variable. Sur un clavier de PC, c'est le fameux « tiret du huit » . Sur un clavier Mac, c'est le tiret en majuscule.

Si tu n'as fait aucune faute de saisie, Python accepte cette instruction sans émettre de message d'erreur. Pourtant, il semble que rien ne se soit passé. Python réaffiche les trois signes de son invite et se met en attente de ta prochaine commande.

Saisis maintenant l'instruction suivante :

```
print ma_varinum
```

Dès que tu valides par la touche Entrée, Python va chercher dans la mémoire s'il existe une variable portant le nom `ma_varinum`, puisque tu l'as indiqué à la suite du mot-clé d'une commande Python, `print`. Il affiche alors la valeur qui avait été stockée dans cette variable.

Mais, puisqu'une variable est variable, nous pouvons donc modifier son contenu, comme ceci :

```
ma_varinum = 2
```

Dès que tu valides par Entrée, si tu redemandes l'affichage du contenu de la variable, tu constates qu'elle contient dorénavant la valeur 2 et non plus la valeur 1. Observe le contenu de la Figure 8.7.

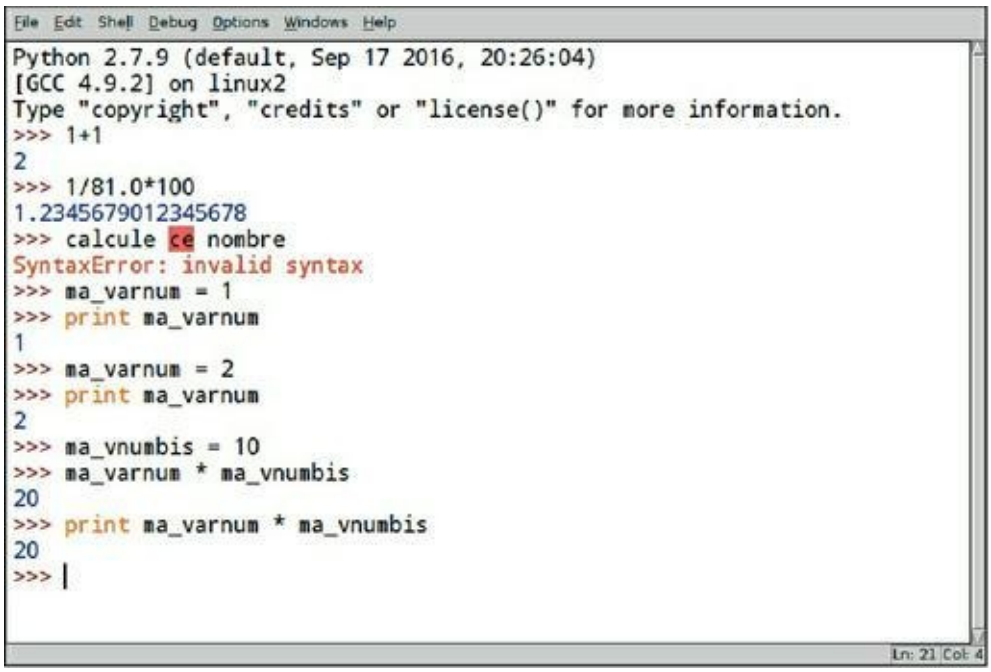


Figure 8.7 : Exemples de commandes en mode interactif.

Le mot anglais `print` signifie « *imprimer* » . Ici, il ne s'agit pas d'envoyer des données à l'imprimante, mais de les afficher sur l'écran, plus exactement, dans la fenêtre de Python. Cette commande n'a aucun effet sur la valeur de la variable. Dans tes projets ultérieurs, tu n'auras généralement pas besoin d'utiliser la commande d'affichage. L'essentiel est de pouvoir utiliser les variables, modifier leur contenu, le relire, bref, s'en servir. Dans la fenêtre que nous utilisons en ce moment, Python affiche de lui-même le résultat de la commande que tu lui fournis. Dans les fenêtres d'édition que nous allons voir d'ici quelques pages, rien ne sera affiché, à moins d'utiliser la commande `print`.

Utiliser une variable

Stocker des valeurs dans des variables permet de conserver ces valeurs, mais tu peux aller plus loin : tu peux directement demander de faire des calculs entre plusieurs variables. Saisis les deux lignes suivantes, sans oublier d'utiliser la touche Entrée à la fin de chacune :

```
ma_vnumbis = 10
ma_varnum * ma_vnumbis
```

Après la première ligne, Python a réservé un nouvel espace en mémoire pour la seconde variable. Après la seconde instruction, Python a effectué un calcul en multipliant le contenu des deux variables.

Pouvoir combiner des noms de variables ouvre un vaste champ de possibilités ! Tu peux par exemple créer une série de variables puis définir une formule pour effectuer la somme de leurs valeurs. Il suffit de changer le contenu d'une variable puis de demander le même recalcul, un peu comme tu peux le faire dans une feuille de calcul de tableur.

Recettes et algorithmes

Tu viens d'acquérir plusieurs superpouvoirs : tu sais réserver des emplacements dans la mémoire, y stocker des valeurs et faire des calculs en combinant des contenus. La prochaine technique fondamentale consiste à stocker le résultat d'un calcul dans une autre variable.

```
ma_vnumgrande = ma_varnum * ma_vnumbis
```

Tu vois apparaître un nouveau symbole, `*`, qui est un opérateur. Dans le langage Python, cet opérateur est celui de la multiplication. La touche correspondante sur ton clavier fait partie du pavé numérique à droite. N'utilise surtout pas un `x` minuscule ou majuscule pour les multiplications : Python va croire que tu lui indiques le nom d'une variable.

Cette unique ligne d'instruction est une véritable recette pour traiter de l'information. Tu n'as plus besoin d'afficher le contenu des deux variables de droite. La seule condition est qu'elles contiennent des données du même type. Ici, il faut que ce soit des valeurs numériques entières. La recette fonctionnera, quelles que soient ces valeurs.

Évidemment, le résultat risque d'être étrange si tu essaies de multiplier des mots, des photos ou des fichiers audio !

Une partie du travail de programmation consiste à écrire ce genre de recette. Les formules mathématiques les plus courantes sont toutes déjà disponibles dans Python sous forme d'opérateurs ou de fonctions. Dès que tu as besoin de ce genre de calcul, il suffit d'utiliser le mot-clé correspondant pour appliquer le calcul à une ou plusieurs variables.

Tu peux évidemment créer tes propres formules pour les appliquer à différents types d'information. Cela permet d'inventer des traitements que tu appliqueras par exemple à des échantillons sonores, à des pages Web, à des images, *etc*.

En jargon informatique, les recettes sont des algorithmes. Ce mot provient du nom d'un célèbre mathématicien arabe du Moyen Âge.

L'interpréteur et l'éditeur

La fenêtre de Python dans laquelle nous travaillons depuis le début du chapitre est la fenêtre de l'interpréteur. Elle est pratique pour faire de petits tests et des calculs.

Mais Python est d'abord et surtout destiné à écrire des programmes comportant plusieurs lignes d'instruction qui ne seront exécutées qu'une fois toutes les lignes écrites et non interprétées l'une après l'autre juste après avoir été saisies. Il faut bien distinguer la fenêtre de l'interpréteur shell et celle que nous allons découvrir maintenant : la fenêtre de l'éditeur.

Dans la fenêtre de l'interpréteur shell, dès que tu frappes la touche Entrée, le programme interpréteur analyse ce qui a été saisi et essaye de traiter la demande en la retransmettant au moteur technique de Python. Ce dernier effectue le traitement et renvoie une réponse. C'est cette réponse qui est affichée lorsque c'est possible dans la fenêtre de l'interpréteur.

Voici les différentes étapes résumées du fonctionnement de l'interpréteur :

1. Affichage des trois signes de l'invite et mise en attente de la saisie d'une commande et de la frappe de la touche Entrée.
2. Analyse de ce qui a été saisi.
3. Vérification de l'absence d'erreur.
4. Si un problème est détecté, affichage d'un message et passage à l'Étape 7. Si tout va bien, envoi du contenu de la ligne au moteur Python et poursuite à l'Étape 5.
5. Attente du renvoi du résultat par le moteur Python.
6. Si le résultat est renvoyé, affichage dans la fenêtre de l'interpréteur.
7. Affichage de l'invite et mise en attente de la prochaine commande.

Tu n'as pas besoin de mémoriser cette liste. Elle ne sert qu'à te montrer que l'interpréteur est un programme. Les programmeurs qui ont conçu l'interpréteur ont écrit cette séquence d'actions pour savoir comment leur projet devait être réalisé. Ils ont ensuite rédigé les instructions pour chacune des étapes.

Le moteur de Python est assez complexe. Il n'a pu être créé qu'après une analyse très poussée. Lorsque tu vas à ton tour créer des programmes, même s'ils sont simples, il est conseillé de prendre la bonne habitude de décomposer les principales étapes du traitement avant de commencer à programmer.

Cette séquence d'étapes est parfois appelée une spécification ou une analyse. Elle décrit les différentes étapes du programme, sans descendre jusqu'au niveau des instructions réelles en langage Python.

Une bonne spécification n'oublie pas de prévoir tous les cas dans lesquels les choses ne se déroulent pas comme prévu. Cela correspond à la gestion des erreurs, qu'elles soient causées par l'utilisateur ou par un résultat intermédiaire imprévu.

La spécification joue un peu le rôle d'une carte d'orientation pour se repérer dans le code. Chaque instruction réalise une opération élémentaire. Une fois que tu réunis un certain nombre d'instructions, tu obtiens un comportement plus complexe. Le résultat est une application, comme celles que tu as sur ton téléphone ou qui animent les sites Web. Même le petit programme qui permet de régler les différents modes de cuisson dans un four à micro-ondes sans mettre le feu à ton domicile est une application.

Découvrir la fenêtre d'édition

Voyons maintenant comment faire pour pouvoir écrire plusieurs lignes d'instruction en vue de les faire exécuter d'une traite. Pour rédiger ce qui va devenir le code source, tu vas utiliser un autre type de fenêtre dans l'environnement Python : la fenêtre d'édition.

Pour ouvrir cette fenêtre d'édition, ouvre le menu `File` et choisis `New file`, comme le montre la Figure 8.8. Au départ, la barre de titre contient la mention `Untitled`, ce qui signifie que le contenu n'a encore jamais été stocké dans un fichier permanent. La fenêtre est totalement vide. Tu remarques qu'il n'y a pas d'invite pour saisir une commande. La barre de menus comporte quelques différences par rapport à la fenêtre principale : il y a une commande `Format` et une commande `Run` au lieu des commandes `Shell` et `Debug`.

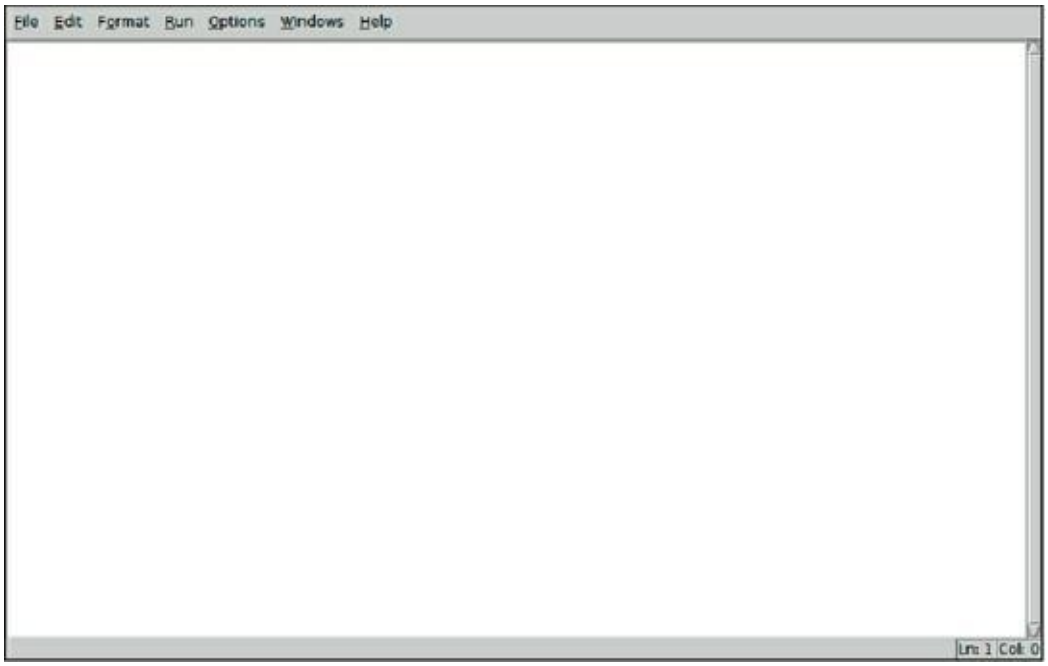


Figure 8.8 : La fenêtre d'édition de Python.

Rédiger du code

Pour rédiger le code source, il suffit de le saisir dans la fenêtre de l'éditeur en utilisant la touche Entrée à la fin de chaque ligne. La différence est que la frappe de Entrée ne provoque pas une tentative d'exécution de la ligne. Tu peux simplement continuer à saisir la ligne suivante.

Pour notre premier projet, nous allons essayer de reproduire l'exemple que nous avions réalisé en mode interactif dans les pages précédentes. N'oublions pas d'utiliser la commande `print` pour pouvoir vérifier le résultat.

Voici les quatre instructions que je te propose de saisir avec grand soin dans la fenêtre d'édition. Elles sont également visibles dans la Figure 8.9.

```
ma_varnum = 1234
ma_vnumbis = 5678
resultat = ma_varnum * ma_vnumbis
```

```
print resultat
```

Une fois que tu as saisi la dernière ligne, tu constates que le mot `print` a été coloré. C'est la preuve qu'il s'agit d'un mot réservé, donc connu par le langage Python. Tes noms de variables te sont personnels. Ils ne sont donc pas colorés.

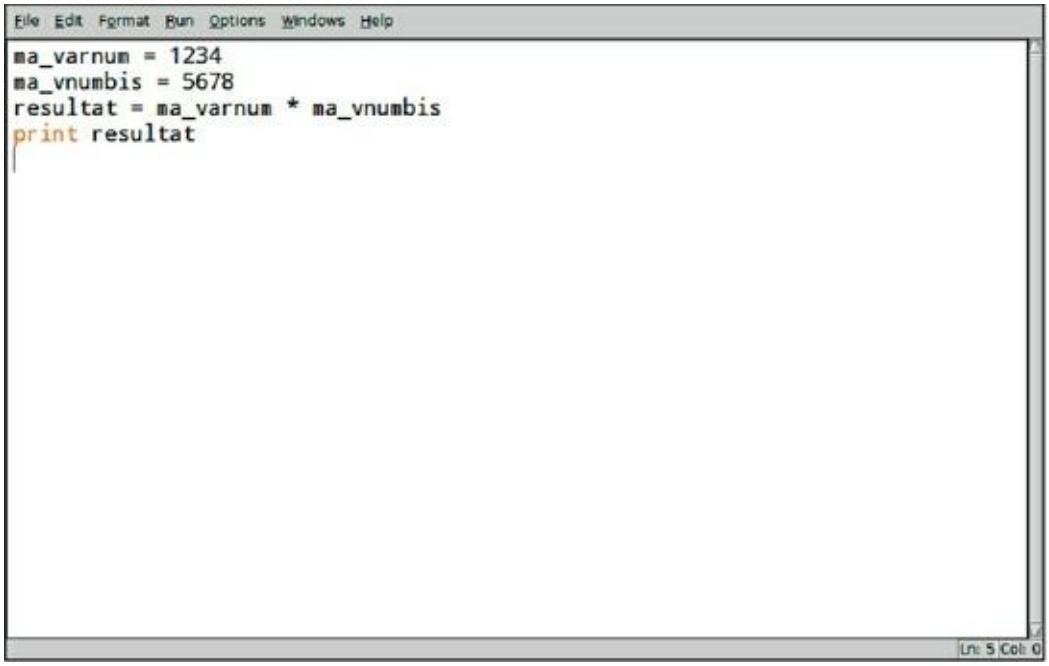


Figure 8.9 : Première séance de rédaction de code source.

Exécuter le code

Une fois que le code est saisi, tout programmeur n'a qu'une envie : vérifier s'il fonctionne. Pour lancer l'exécution du contenu de la fenêtre d'édition, ouvre le menu `Run` et choisis la commande `Run module`. Tu vas rapidement prendre l'habitude d'utiliser plutôt le raccourci clavier qui est la touche F5.

Mais pas de précipitation. Python va refuser pour l'instant de lancer l'exécution. Il se doute que tu voudras pouvoir conserver ce code source pour ne pas avoir à le ressaisir un prochain jour. Il te demande donc par une boîte d'alerte si tu acceptes de sauvegarder ce code

source dans un fichier. Clique OK dans cette boîte pour accéder à la boîte d'enregistrement de fichiers (Figure 8.10).

Figure 8.10 : La boîte d'enregistrement de fichiers sources.

Lors de l'ouverture de la boîte, le répertoire/dossier proposé est ton dossier personnel dans Linux. Si tu as ouvert la session avec le nom et le mot de passe proposés au départ, cela correspond au sous-dossier `/home/pi`.

Tu peux stocker le premier programme dans ce dossier. Clique dans la zone de saisie du nom, `Filename`, et saisis par exemple le nom suivant :

```
projet_1.py
```

Clique ensuite le bouton `Save`.

Prends soin de choisir des noms significatifs pour les fichiers de tes projets. Si tu choisis par exemple un nom constitué seulement de la lettre `a`, tu auras bien du mal à savoir à quoi correspond ce programme si tu y reviens dans plusieurs mois. Tu remarques au passage que tous les fichiers sources de Python doivent se terminer par une extension `.py`.

La boîte d'enregistrement n'offre pas de commande pour créer directement un dossier, ce qui est dommage, car tu auras certainement envie de regrouper tous tes projets Python dans un même dossier. La solution est simple : il suffit d'annuler l'enregistrement dans l'éditeur

de Python puis d'ouvrir une fenêtre du Gestionnaire de fichiers. Dans cet outil, tu peux créer un nouveau dossier que tu appelles par exemple `Projets_Python` directement dans ton répertoire personnel `home`.

J'en profite pour t'inviter à ne jamais utiliser d'espace dans les noms des dossiers, ni dans les noms des fichiers. Dès que tu as besoin d'ajouter une espace pour faciliter la lecture, utilise un caractère de soulignement, comme dans mon_nom_de_projet.

Dès que tu valides la saisie, Python affiche une fenêtre d'interpréteur avec la mention RESTART. S'il n'y a pas d'erreur dans ta saisie, tu dois voir apparaître le résultat numérique du calcul. La valeur exacte dépend de celle que tu as choisi d'indiquer pour les deux variables.

En cas d'erreur

Si une erreur est détectée, repasse dans la fenêtre d'édition et relis le code soigneusement. Tu as peut-être oublié un caractère de soulignement ou utilisé le tiret par mégarde. Vérifie bien le nom de tes variables. Tu n'as pas utilisé la lettre X à la place de la barre oblique pour l'opérateur de multiplication ? Tu n'as pas non plus écrit toutes les instructions sur une seule ligne ?

En effet, les ordinateurs sont très pointilleux. Tu peux donner libre cours à ta fantaisie pour le choix des noms de tes variables, mais il faut tenir compte de quelques contraintes ici aussi, et notamment pas de lettres accentuées, comme tu peux le voir dans le nom `resultat`.

Nous en savons maintenant assez pour nous attaquer à un véritable projet de programmation, ce qui est le but du prochain chapitre.

Chapitre 9
Python le devin

Dans le Chapitre 8, nous avons fait nos premiers pas dans l'environnement de programmation du langage Python. Voyons maintenant comment mettre à profit tout cela en créant un projet complet. Notre programme va essayer de deviner un nombre entre 1 et 10 choisi par le joueur.

Cela semble facile, mais ce n'est qu'une apparence. Il faut en effet prévoir tout ce qui doit pouvoir se produire au cours du jeu. Cela suppose aussi de savoir un peu mieux comment fonctionne le langage Python.

Réfléchir avant d'écrire

Est-ce que l'on peut dire que quelqu'un est devenu un bon programmeur à partir du moment où il n'a plus besoin de réfléchir avant de commencer à écrire son programme ?

Absolument pas !

Réfléchir avant de commencer à coder est indispensable et les meilleurs programmeurs font beaucoup d'analyse avant d'écrire la première ligne de leur projet. Ils commencent par subdiviser le projet en sous-ensembles, puis en étapes élémentaires.

Un programmeur doit sans cesse apprendre de nouvelles techniques. C'est le contraire de ce que l'on croit, et de ce qui est la règle dans d'autres métiers. Le programmeur ne peut pas commencer par apprendre en une fois toutes les techniques puis les appliquer tout au long de son existence.

La vie du programmeur est faite de découvertes incessantes. Parfois, il s'agit de comprendre comment réaliser quelque chose dans un nouveau langage, comme par exemple Python. Parfois, cela suppose de chercher s'il n'y a pas un autre programmeur qui a déjà eu le même problème et qui a trouvé une solution. Parfois, il s'agit d'étudier le code source d'un autre projet pour voir si on ne pourrait pas réutiliser une technique dans un nouveau domaine.

Il est donc tout à fait normal de ne pas savoir par où commencer. Il en va ainsi dans pratiquement tous les projets informatiques.

Savoir ce que l'on doit apprendre

Pour le projet de ce chapitre, nous devons commencer par définir les grandes lignes du fonctionnement, sans nous noyer dans les détails.

Cette étape est assez facile. Quasiment tous les programmes informatiques suivent la même logique. Voici un exemple :

1. Nous commençons par poser une question à l'utilisateur/joueur, puis attendons qu'il réagisse en cliquant un bouton (ou alors nous lisons le contenu d'un fichier).
2. Si l'utilisateur a saisi quelque chose, nous vérifions que cette saisie est exploitable.
3. Nous pouvons alors réaliser une action ou un traitement.

   Dans le cas de notre jeu, nous devons écrire les instructions pour deviner un nombre.
4. Nous affichons alors notre réponse (ou nous la stockons dans un fichier).
5. Nous vérifions que nous n'avons plus rien à faire.

6. Si l'utilisateur veut arrêter, le programme se termine. Dans le cas contraire, nous recommençons à l'Étape 1.

Dans les jeux, le programme se place généralement en pause en attendant que l'utilisateur clique un bouton ou frappe une touche. Une fois l'événement détecté, le jeu réagit par exemple en changeant la position d'un vaisseau spatial, puis en mettant à jour le score (sauf si le vaisseau explose, bien sûr). Dans tous les cas, le programme produit des sons et musiques différents selon que le joueur a gagné ou perdu.

En programmation, les informations qui viennent du monde extérieur s'appellent les entrées, *input*. Le programme travaille sur ces données pour produire des résultats qui s'appellent des sorties, *output*. Le travail réalisé sur ce qui est reçu en entrée s'appelle le traitement des données. La séquence générale d'un programme consiste donc en trois étapes : entrées, traitements, sorties.

La décomposition faite ci-dessus permet de dessiner un logigramme comme celui de la Figure 9.1. À partir de là, tu peux dresser la liste de ce que tu as besoin d'apprendre à faire. Commence par essayer de trouver tout seul, puis compare avec ma liste :

- Comment puis-je poser une question en Python ?
- Comment puis-je récupérer la réponse saisie au clavier ?
- Comment vérifier que la réponse est compréhensible ?
- Comment puis-je deviner un nombre de façon intelligente ?
- Comment puis-je savoir que j'ai terminé ?
- Comment puis-je revenir au début de mon programme ?

» Comment puis-je arrêter le programme ?

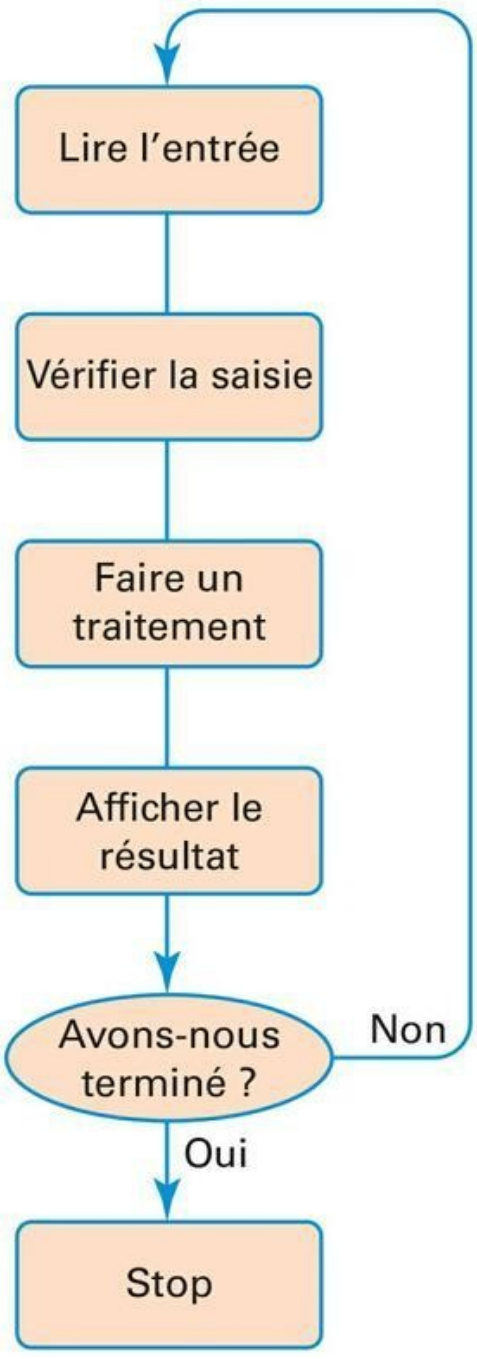


Figure 9.1 : Le logigramme de notre projet Python.

Quelle ribambelle de questions pour un programme aussi simple !

Mais pas de panique. Tu n'as pas besoin de trouver toutes les réponses avant de commencer. Nous allons répondre à nos besoins un par un. D'une mission impossible, nous ferons une séquence de travaux réalisables.

Le seul moyen d'y parvenir consiste à se dire que la question initiale, créer un jeu vidéo sympa, doit se décomposer en un certain nombre de problèmes plus simples, comme par exemple comment arrêter le jeu. Si tu tentes de plonger dans l'écriture du programme sans le décomposer d'abord, c'est un peu comme si tu sautais dans une fosse aux crocodiles avec une main attachée dans le dos et l'autre qui doit tenir la souris.

En travaillant étape par étape, tu vas pouvoir rédiger les instructions qui répondent à chaque problème en particulier, sans te soucier des autres problèmes.

Certains livres de programmation commencent par présenter la totalité du code source en te demandant de le saisir sans réfléchir, puis se contentent de quelques explications. C'est une approche séduisante en apparence, mais elle ne permet pas d'apprendre de façon solide. Commencer par se poser des questions semble moins amusant, mais une fois que tu auras passé cette première étape d'analyse, tu sauras écrire les instructions de façon bien plus efficace, et tu risqueras bien moins d'être bloqué en plein milieu de ta progression.

Questionner le joueur

Comment faire pour poser une question au joueur de ton jeu en langage Python ? Où trouver des informations ?

Sur Internet, tu peux trouver des idées en cherchant par exemple « Python saisie » . Voici une page du tutoriel de référence de l'association française des utilisateurs de Python (afpy.org) :

En faisant des recherches sur Internet, tu vas trouver des centaines de sites, la plupart en anglais, à commencer par le site officiel de Python, python.org. En français, tu pourras visiter le site wiki.python.org.

Pour l’instant, ne perds pas de temps à te plonger dans ces sites, car tu risquerais de te noyer dans les détails.

Réalisons d’abord le projet de ce chapitre.

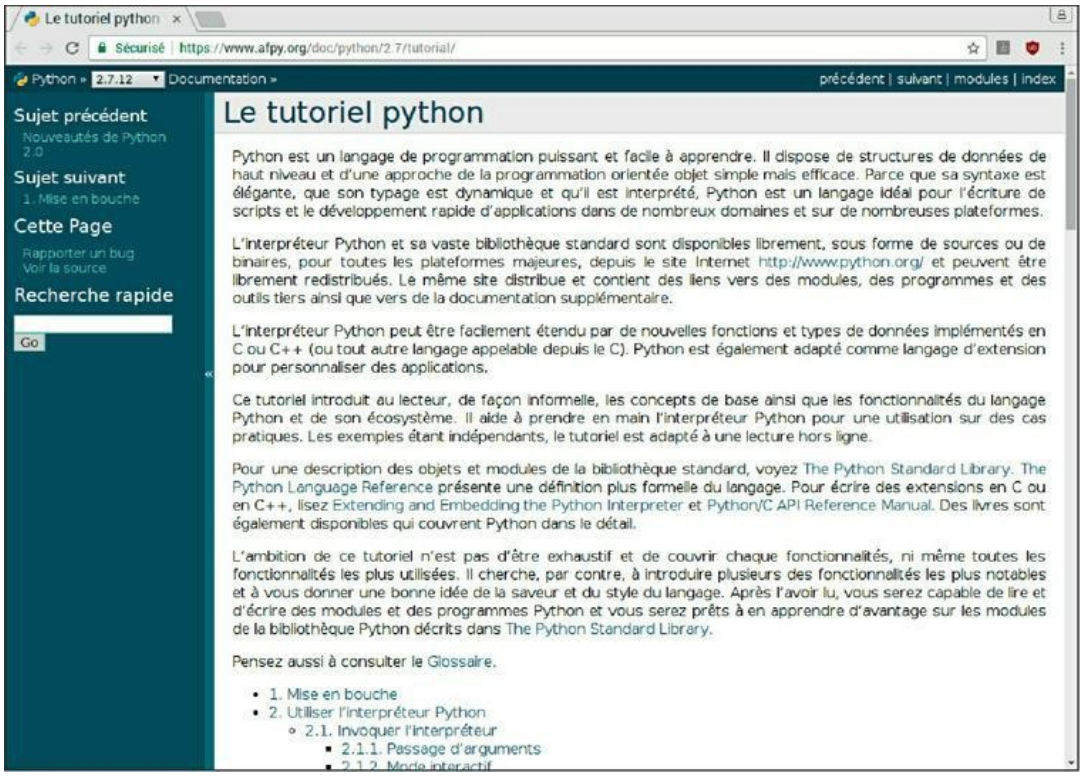


Figure 9.2 : Le tutoriel Python du site afpy.org.

La fonction raw_input()

Ce projet est petit, mais il est interactif. Nous allons d’abord écrire le code qui va permettre de capturer les touches frappées par le joueur en réponse à notre question. Nous allons utiliser pour ce faire une fonction prédéfinie dans le langage Python. Nous ne voulons récupérer cette entrée que telle qu’elle a été saisie, sans essayer aucun traitement intermédiaire.

Comme une entrée se dit *input* en anglais, le mot-clé va comporter le mot `input`. Nous voulons récupérer l’entrée sous sa forme brute, ce qui se dit en anglais *raw*. La nouvelle fonction s’appelle justement `raw_input()`.

Il s’agit d’une fonction, c’est-à-dire d’un sous-programme dont tu n’as pas besoin d’écrire le code source. Il existe déjà. Pour s’en servir, tu indiques son nom puis une paire de parenthèses. Entre ces parenthèses, tu indiques une valeur à transmettre au démarrage de la fonction, ce qui se dit un argument ou paramètre.

Comme quasiment toutes les fonctions, dans tous les langages, une fois l’exécution terminée, la fonction va renvoyer une valeur qui va apparaître pour l’ordinateur à la place du nom de la fonction. Il nous faut donc ensuite capturer cette valeur en la stockant dans une variable.

Cette copie d’une valeur vers une variable utilise un opérateur qui est le signe égal (=). Il n’a pas du tout le même sens que dans la vie courante. Il ne veut pas dire que ce qui est à gauche est égal à ce qui est à droite ; il signifie « Copier la valeur qui est à droite du signe dans l’emplacement mémoire symbolisé par le nom indiqué à gauche du signe » . Voici donc cette unique ligne d’instruction qui a réclamé une aussi longue explication :

```
sReponse = raw_input("C’est oui ou c’est non ?" )
```

La variable que nous avons choisi de nommer `sReponse` va contenir la valeur qui a été lue au clavier par la fonction `raw_input()`une fois que cette ligne d’instruction aura été exécutée.

Mise en pratique

Il est toujours conseillé de tester chaque nouvelle technique juste après l’avoir apprise. Nous allons créer un petit projet parallèle à cet effet.

Voici comment tester notre code de saisie de frappe utilisateur :

1. Dans la fenêtre principale de l’interpréteur, choisis la commande `File/New file` pour ouvrir une fenêtre

d’édition.

2. Dans cette fenêtre d’édition vide, saisis soigneusement les deux lignes d’instruction suivantes :

Listing 9.1 : Code source de devinet_1.py

```
sReponse = raw_input("Saisis oui ou non : ")
print "Tu as repondu : " + sReponse
```

3. Stocke le projet dans un fichier en ouvrant le menu `File` et en choisissant `Save`. Comme nom de fichier, choisis par exemple ceci :

`devinet_1.py`

4. Lance l’exécution du programme par la touche F5 ou la commande `Run/Run module`.

   Si tu n’as fait aucune faute de saisie, tu dois voir un résultat proche de celui de la Figure 9.3.

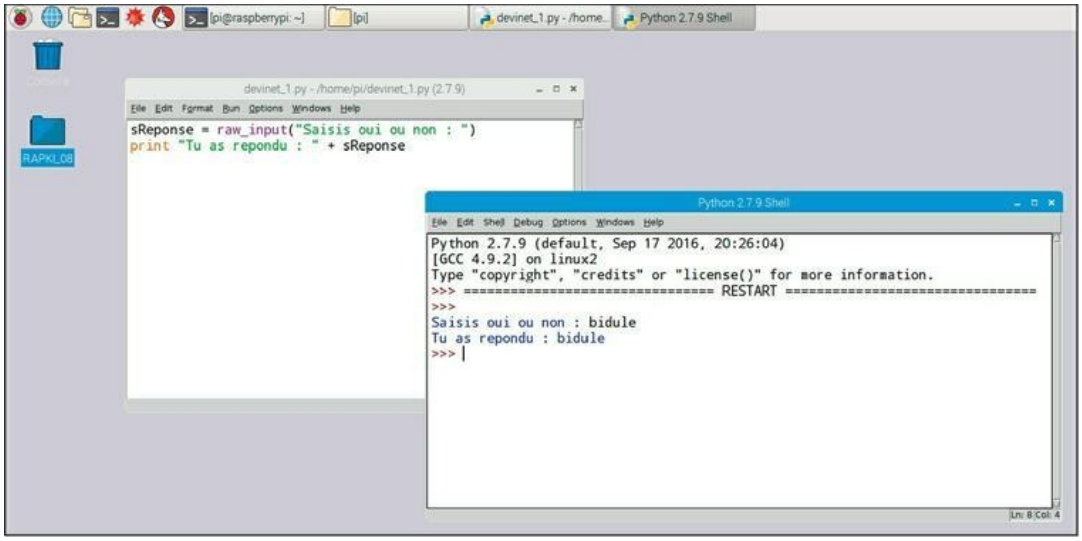


Figure 9.3 : Exécution du programme devinet_1.py.

N. d. T : Tu as remarqué que j'ai évité d'utiliser les lettres accentuées. Si tu as saisi un accent pour « Tu as repondu » , tu auras certainement vu apparaître un message d'erreur te proposant d'accepter de basculer dans le codage UTF-8. Dans ce cas, tu peux répondre OK ou bien `Edit my file`, ce qui rajoute une directive tout au début du code source. Je ne vais pas aller plus loin ici dans l'explication de ces conventions. Quelle que soit la réponse, le programme acceptera ensuite les lettres accentuées.

Attention : les lettres accentuées ne seront jamais acceptées dans les noms de variables et de fonctions.

La fonction `raw_input()` ne cherche pas à vérifier l'exactitude de ce qui a été saisi. Elle est volontairement « brute » . Si tu saisis n'importe quoi, ce même n'importe quoi sera affiché par l'instruction `print`, puisque c'est le contenu de la variable.

Avec ces deux simples lignes, nous avons déjà répondu à deux des questions de notre liste de choses à faire : nous savons comment poser une question à l'utilisateur et comment récupérer sa réponse.

C'est maintenant que les choses deviennent plus délicates : comment savoir si la réponse peut être utilisable par le programme ?

Cette façon de progresser t'aide à comprendre pourquoi il est important de subdiviser chaque problème en plusieurs étapes élémentaires. Tu peux ainsi avancer par petits pas assurés.

Dans la seconde instruction, tu as sans doute remarqué que nous utilisons le signe +. C'est un opérateur qui sert normalement à additionner deux valeurs numériques. Comment peut-on s'en servir pour additionner deux messages ? En fait, la commande d'affichage sait rabouter un texte à la suite d'un autre au moyen de cet opérateur +. Cette opération porte le nom de concaténation, c'est-à-dire d'ajout d'une chaîne de caractères à la suite d'une autre (du latin *catena* qui signifie « chaîne »). Une suite de lettres de l'alphabet et de signes s'appelle une chaîne de caractères, en anglais *string*.

Tester la réponse

Pour savoir si nous pouvons utiliser la réponse saisie, il nous faut pouvoir distinguer entre une bonne et une mauvaise réponse.

Notre jeu de devinette est extrêmement simple : nous n'avons besoin que de deux réponses, soit « oui » , soit « non » . Dès que le joueur saisit autre chose que l'un de ces deux mots, nous ignorons la saisie.

Mais nous ne pouvons pas nous contenter d'ignorer la saisie et d'arrêter de jouer. Si la saisie n'est pas utilisable, il faut poser à nouveau la question, et cela sans cesse jusqu'à ce que le joueur saisisse soit le mot « oui » , soit le mot « non » . Il nous faut donc apprendre comment faire un test en Python et comment mettre en place une boucle de répétition.

En fait, nous avons remplacé notre problème unique (comment vérifier que la saisie est utilisable ?) par deux problèmes :

- Comment vérifier que la réponse est soit oui, soit non ?
- Comment continuer à poser la question jusqu'à ce que la réponse soit utilisable ?

Une fois de plus, nous avons subdivisé un problème en deux problèmes plus simples. Quand tu programmes, cela t'arrivera souvent de voir la liste des choses restant à faire grossir pendant un moment, mais je te promets qu'elle finira par se vider totalement !

Dans un grand projet, il faut continuer à subdiviser les problèmes sans cesse. Par exemple, les jeux vidéo comme *Angry Birds*, *Warcraft* ou *GTA* représentent la solution à des milliers de problèmes ; il y en a tant qu'il est impossible à une seule personne de créer un tel projet. Voilà pourquoi ce sont des équipes de développeurs qui y travaillent pendant des années. La question d'échelle (et de budget) mise à part, ils font la même chose que nous ici : subdiviser les gros problèmes en petits problèmes puis écrire le code qui correspond à chaque solution.

Pour tester une valeur

Il y a deux grandes techniques en informatique pour tester si une valeur correspond à une condition ou pas. La première technique consiste à utiliser le test conditionnel basé sur le mot réservé if :

```
if condition_vraie
   faire un traitement
else
  faire un autre traitement
```

L'autre technique consiste à utiliser un bloc de répétition conditionnelle qui consiste à répéter une ou plusieurs actions tant qu'une condition n'est pas encore satisfaite :

```
repeat
  faire un traitement
until
  condition_devenue_vraie
```

L'exécution conditionnelle utilisant le mot-clé if (qui signifie « si ») ne répond à notre problème qu'en partie. Ce test permet de savoir si la condition est remplie, c'est-à-dire si l'utilisateur a saisi soit « oui » , soit « non » , mais cela ne permet pas de reposer la question si l'utilisateur a saisi autre chose.

En revanche, la répétition conditionnelle répond aux deux besoins. Elle permet de tester la saisie et de reboucler si la réponse n'est pas l'une des deux attendues :

```
repeat
  sReponse = raw_input("Saisis oui ou non :
")
until
  sReponse = "oui" or sReponse = "non"
```

Autrement dit, nous reposons la question tant que ce n'est pas une des deux réponses attendues. Cela semble parfait !

Hélas non. Le langage Python ne connaît pas la technique de répétition repeat..until, à la différence d'autres infolangages. Nous devons trouver une autre solution.

D'abord une petite remarque capitale. Que se passe-t-il lorsque l'on écrit ce genre d'instruction, comme dans la quatrième ligne ci-dessus :

```
sReponse = "oui"
```

En fait, c'est une terrible erreur de logique. Nous ne faisons que stocker la valeur littérale « oui » dans la variable portant le nom sReponse. Ce n'est pas ainsi que l'on teste si cette valeur est identique au contenu de la variable.

Le signe égal isolé = est un opérateur pour copier une valeur, pour l'affecter à une variable. Pour comparer, il faut utiliser le double signe égal, ==.

Si tu as de toi-même remarqué cette erreur, tu es peut-être un génie en herbe de la programmation. Si tu n'avais rien remarqué, ne t'inquiète pas. La plupart des gens font cette erreur au début, et même des développeurs chevronnés confondent parfois les deux opérateurs. Voyons donc celui dont nous avons besoin.

L'opérateur de comparaison ==

Pour savoir si deux choses sont identiques, il faut utiliser le double signe ==. Voici notre instruction reformulée :

```
repeat
  sReponse = raw_input("Saisis oui ou non :
")
```

```
until
  sReponse == "oui" or sReponse == "non"
```

Pour bien mémoriser cette distinction entre l'opérateur d'affectation constitué d'un seul signe égal et l'opérateur de comparaison constitué de 2 signes égal, relis sept fois cette phrase.

Se répéter avec while

Nous n'avons résolu que la moitié du problème. Quel mot-clé offre le langage Python pour répéter une ou plusieurs instructions jusqu'à ce qu'une condition change ?

Ce mot-clé s'écrit while, qui signifie « tant que » . Voici comment l'utiliser :

```
while expression_de_test :
  première instruction conditionnelle
  autre instruction conditionnelle
(suite du programme après la boucle)
```

L'expression de test doit être vraie pour que les instructions qui sont contrôlées par le mot-clé soient exécutées. Dès que le test n'est plus vrai, on sort de la boucle de répétition et on poursuit à l'instruction suivante.

Aurions-nous ainsi résolu notre problème ? Pas entièrement.

Tester la condition inverse

Une devinette pour toi : quelle est la condition à écrire pour que l'on répète la demande de saisie à l'utilisateur ? Autrement dit, que faut-il écrire à droite du mot-clé while ?

Lis le contenu du Tableau 9.1 pour t'aider dans ta réflexion.

Tableau 9.1 : Les conditions de rebouclage.

« oui » a été saisi	*« non » a été saisi*	*Reboucler ?*
Affirmatif	Qu'importe	Inutile
Qu'importe	Affirmatif	Inutile
Négatif	Négatif	Absolument !

Nous devons reposer la question au joueur si sa dernière réponse n'est ni égale à « oui » , ni égale à « non » .

Ce double test négatif peut sembler difficile à comprendre à première vue. Ne t'inquiète pas si tu te crois un peu perdu. Continue la lecture. Le code source va tout éclaircir.

Nous devons d'abord découvrir un nouvel opérateur qui permet de vérifier qu'une donnée est différente d'une autre. Ce nouvel opérateur combine deux symboles différents comme ceci :

```
while  sReponse != «oui»
```

L'opérateur est constitué d'un signe point d'exclamation suivi d'un signe égal. Tout ce qui suit le mot-clé while constitue l'expression dont le résultat final doit être vrai. Dans cette instruction, notre condition est vraie lorsque la variable `sReponse` ne contient pas la chaîne « oui » , ce qui est bien ce que nous désirons.

Ajoutons donc une boucle de répétition while à notre version actuelle du projet. Le plus simple consiste à partir de la première version en procédant immédiatement à son enregistrement sous un nouveau nom. Utilise la commande `File/Save as` et enregistre la nouvelle version sous le nom

`devinet_2.py`.

Place ensuite le curseur à la fin de la première ligne et insère une ligne vide avec Entrée. Remonte au début de la nouvelle première ligne et saisis la deuxième ligne de l'exemple complet qui suit puis la troisième ligne avec l'indentation qui va être insérée automatiquement par Entrée.

Listing 9.2 : Code source de devinet_2.py

```
sReponse = raw_input("Saisis oui ou non : ")
while sReponse != “oui” and sReponse !=
"non":
    sReponse = raw_input("Saisis oui ou non
: ")
print "Tu as repondu : " + sReponse
```

Nous avons maintenu en place la première ligne de la première version parce qu'il faut stocker une première fois le résultat de la saisie dans notre variable avant d'entrer dans la boucle. Nous aurions pu également stocker au départ dans `sReponse` quelques espaces ou bien un autre mot, afin de garantir que nous poserons au moins une fois la question en faisant échouer le test.

Dans la documentation Python du mot-clé `while`, il est indiqué que sa condition doit être vraie, en indiquant `while True:`. Ici, nous combinons deux expressions de test avec l'opérateur `and`. Autrement dit, le test qui suit le mot `while` peut se comprendre de la manière suivante :

```
tant que sReponse ne contient pas "oui" ET
que sReponse ne contient pas
"non"
  ...
```

Le signe : et les indentations

Tu as peut-être constaté la présence du signe : (deux-points) à la fin de l'expression de test. Ce signe est absolument indispensable. Il permet au langage Python de savoir que ce qui le suit est un bloc d'une ou de plusieurs lignes d'instructions qui dépendent du mot-clé while. Ces lignes ne seront exécutées que si la condition du while est satisfaite.

Lorsque tu as saisi le signe : puis frappé la touche Entrée, tu as constaté que la ligne suivante a automatiquement été indentée de quatre espaces vers la droite. Dans le langage Python, cette indentation n'a pas qu'un objectif esthétique pour améliorer la lisibilité. Dans d'autres langages tels que le langage C, les espaces ne sont pas indispensables, car il faut commencer le bloc conditionnel par une accolade ouvrante {et le refermer par une accolade fermante }. Rien de tout cela en Python.

Si tu enlèves les quatre espaces au début de la troisième ligne, c'est comme si tu avais enlevé les traitements dans la boucle, qui serait devenue vide. La question ne te serait posée que deux fois.

Une fois que tu as saisi les quatre lignes puis enregistré le fichier, tu peux lancer son exécution. Le résultat devrait ressembler à l'exemple de la Figure 9.4. N'hésite pas à saisir plusieurs fois n'importe quoi avant de saisir l'une des deux bonnes réponses.

Le programme ne se termine que si tu saisis « oui » ou « non » (sans les guillemets, bien sûr !).

Notre projet avance, mais il nous a fallu découvrir plusieurs concepts fondamentaux de programmation, n'est-ce pas ?

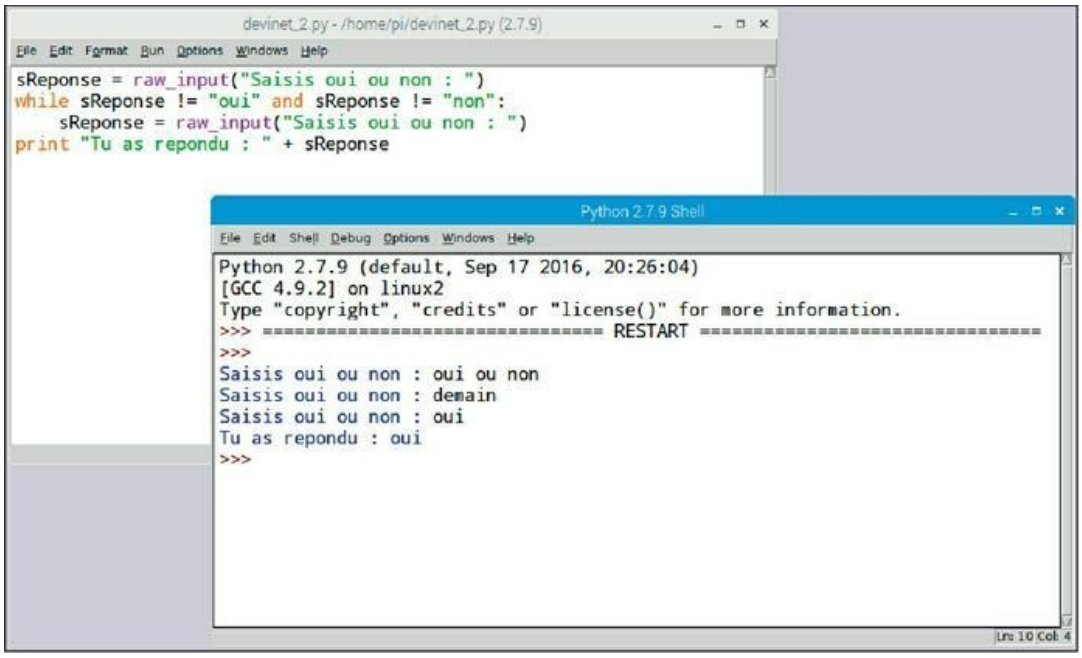


Figure 9.4 : Exécution de devinet_2.py.

Tu auras souvent à écrire du code pour vérifier ce qui a été saisi par un humain. En effet, les humains sont imprévisibles. Il y a tellement de possibilités de se tromper qu'il faut prévoir le plus de cas possibles, tous si possible.

Certains humains peuvent également être malveillants. Les sites Web les plus fréquentés, notamment les sites de commerce et ceux des banques, doivent être très attentifs à ce qu'ils laissent saisir à leurs visiteurs. Un pirate intelligent pourrait se servir de ce moyen d'accès pour récupérer les numéros des cartes bancaires puis s'en donner à cœur joie en achetant tout ce qu'il veut. C'est pour cette raison que nous utilisons la fonction de saisie `raw_input()` et pas sa collègue qui s'écrit `input()`. Cette dernière laisse en effet la possibilité d'envoyer par la saisie des commandes à destination de Python, ce qui peut être très dangereux.

Dans la version 3 de Python, la fonction `raw_input()` porte dorénavant le nom `input()`, et sa collègue dangereuse n'existe plus.

Vous pouvez répéter la question ?

Notre projet est suffisamment avancé pour que nous puissions en réaliser une version complète simplifiée. Pour proposer un nombre et demander au joueur si ce nombre est celui qu'il avait choisi, nous pourrions enchaîner une litanie d'instructions :

```
Ton nombre est-il le 1, oui ou non ?
Ton nombre est-il le 2, oui ou non ?
Ton nombre est-il le 3, oui ou non ?
Etc.
Ton nombre est-il le 10, oui ou non ?
```

Il n'y a pas de magie dans cette version du programme puisque nous finissons toujours par trouver le nombre qu'avait choisi le joueur, sauf s'il triche en ne répondant pas oui au moment approprié.

Cela dit, répéter dix fois quasiment la même instruction n'est pas très efficace. Considère cette préversion comme une expérience personnelle que tu ne montres à personne. Nous allons trouver une solution technique plus satisfaisante dont tu pourras être fier.

Techniques de comptage

Pour compter de 1 à 10 en Python, il y a plusieurs techniques. Nous pourrions ajouter une autre boucle de répétition pour augmenter la valeur d'une variable de 1 tant que cette valeur est inférieure à 11.

Mais si tu as fait quelques essais avec l'atelier Scratch, tu sais qu'il existe une autre boucle de répétition qui porte le nom for. Cette boucle est souvent plus pratique, car elle gère directement une variable qui sert de compteur de tours. La variable qui sert de compteur évolue automatiquement. À chaque tour de boucle for, elle augmente de 1 ou bien elle prend la valeur de l'élément suivant dans une liste.

Tous à la plage : la liste `range` dans Python

Pour contrôler le nombre de tours de la boucle `for`, nous pouvons exploiter un réservoir de valeurs en utilisant la fonction `range` qui renvoie la liste des éléments correspondants. Voici par exemple comment obtenir la liste des dix chiffres de 1 à 10 :

```
range(1, 11)
```

Tu vois que nous utilisons la valeur 11 comme borne supérieure, tout simplement parce que c'est une convention dans le langage Python. La dernière valeur indiquée dans une liste ne fait jamais partie du résultat de cette liste. Il faut le savoir.

Essayons immédiatement de créer une boucle de répétition dans un nouveau petit projet. Ouvre une nouvelle fenêtre d'édition puis saisis les deux lignes suivantes, sans oublier le signe deux-points à la fin de la première ligne :

Listing 9.3 : Code source de compteur_1.py

```
for cpt in range(1,11):
    print cpt
```

Ce bloc signifie « afficher la valeur de la variable `cpt` pour toutes les valeurs entières que l'on trouve dans la plage de valeurs entre 1 et 11, borne supérieure non comprise » .

Autrement dit, cette instruction va véritablement répéter 10 fois l'instruction d'affichage, comme le montre la Figure 9.5. N'hésite pas à sauvegarder puis essayer d'exécuter ce petit projet.

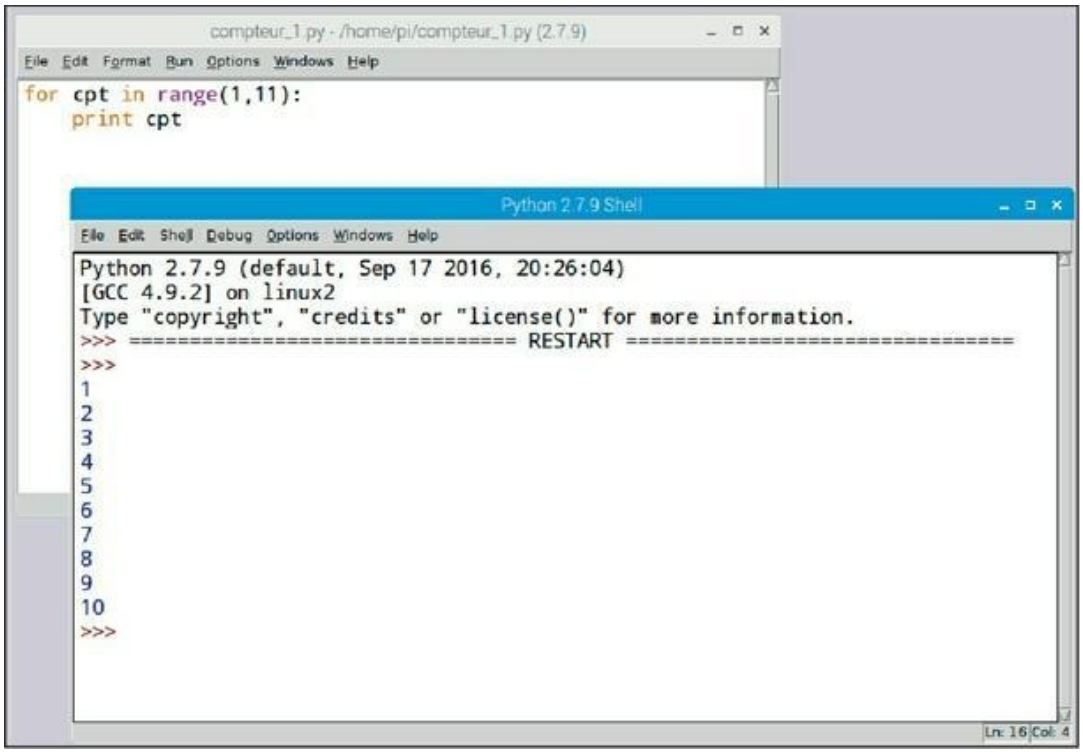


Figure 9.5 : Exécution du projet compteur_1.py.

Dans tous les langages informatiques, certaines techniques sont moins évidentes à maîtriser que d'autres. C'est également le cas en Python, même si ce langage est généralement reconnu comme plus abordable que d'autres. Il faut s'y faire, un point c'est tout.

Une sortie de boucle précoce (break)

Puisque nous en sommes aux boucles for, j'en profite pour te montrer comment sortir d'une boucle avant d'avoir atteint la borne supérieure, ce qui est parfois utile. Nous n'aurons pas besoin de cette technique dans notre projet.

Pour sortir d'office d'une boucle for, il suffit d'utiliser le mot-clé break, qui veut dire rupture :

```
for cpt in range(1,11):
    print cpt
    if cpt == 4 :

        break
```

Dans cet exemple, une fois que le compteur possède la valeur 4, la condition est satisfaite et l'exécution du break fait sortir de la boucle pour continuer avec la suite du programme.

C'est un peu comme un freinage d'urgence, sauf que cela ne sentira normalement pas le caoutchouc brûlé.

Type Torchon et type Serviette

J'ai dit plus haut que tu ne pouvais pas mélanger du texte avec une valeur numérique. Tu ne peux pas non plus combiner deux valeurs numériques pour obtenir une phrase. Autrement dit, les variables ne sont pas toutes du même type. Tu peux le vérifier en réalisant l'exemple suivant dans une nouvelle fenêtre d'édition. Sauvegarde-le sous le nom `compteur_2.py`:

Listing 9.4 : Code source de compteur_2.py

```
for cpt in range(1,11):
    print "Le nombre est : " + cpt
```

La commande d'affichage `print` fait ce qu'elle peut pour deviner le type de ce que tu veux lui faire afficher. C'est pour cela que l'on peut utiliser `print` avec une valeur numérique ou avec du texte.

En revanche, Python n'apprécie pas du tout le mélange des genres. Tu ne peux pas combiner dans le même affichage une valeur du type texte et une valeur numérique avec l'opérateur +. Essaie de faire

exécuter l'exemple précédent. Normalement, tu dois obtenir un message d'erreur dans le style suivant :

```
Type error: cannot concatenate 'str' and
'int' objects
```

Ce message signifie que tu ne peux pas ajouter à la fin d'une variable de type chaîne (str) une variable numérique entière (int).

Allons un peu plus loin. Tous les nombres ne sont pas identiques pour Python. Il y a les nombres entiers et les autres. Les nombres entiers sont ceux qui n'ont pas de chiffres derrière la virgule.

N.D.T. : Dans tes programmes Python, tu ne dois pas utiliser la virgule comme séparateur décimal, mais le signe point, comme dans les pays anglophones.

Dès que tu as besoin de faire des calculs précis, tu utilises des nombres réels que Python reconnaît à la présence d'un point décimal, comme dans `23.205`.

En dehors des deux types numériques, l'autre type très utilisé est le type texte qui correspond à la chaîne de caractères. Une chaîne est une suite de lettres de l'alphabet, de symboles de ponctuation, mais aussi de chiffres, bref, quasiment tout ce que tu peux saisir au clavier.

Si tu définis une chaîne de caractères contenant « 5.1234 » , pour Python, il ne s'agit pas d'une valeur numérique, mais d'une séquence de caractères, équivalente pour lui à « Bonjour » . Voilà pourquoi il n'est pas possible de faire des calculs avec ce type de valeur. Ce sont des chiffres lisibles par les humains, pas des nombres utilisables dans des calculs.

Ces trois catégories de données sont reconnues dans le langage Python par trois mots-clés différents, comme le montrent les tableaux 9.2 et 9.3.

Tableau 9.2 : Les trois types de variables essentiels.

Type	*Exemple*	*Utilisation*

int	10	Nombres entiers
float	3.1459	Nombres réels
str	«bonjour»	Lettres, chiffres et symboles

Comme nous l'avons vu en début de chapitre, tu ne peux pas effectuer des opérations arithmétiques sur des chaînes, mais tu peux malgré tout utiliser l'opérateur + pour coller une chaîne à la suite d'une autre.

Le mot-clé int est l'abréviation du mot *integer* qui signifie en anglais entier. Le mot float signifie que la virgule (le point en fait) est flottante. C'est ainsi que l'on parle des valeurs numériques non entières. Le mot-clé str est l'abréviation de *string*, qui signifie chaîne de caractères. Dans une chaîne, les caractères s'enchaînent les uns à la suite des autres.

Pour faire des calculs mathématiques, tu utiliseras toujours de préférence le type float. Le type int ne fonctionne pas toujours correctement. Dès que tu divises un entier par un autre, mais que le résultat ne tombe pas juste, tu perds les chiffres après le point décimal !

Tableau 9.3 : Exemples de types et d'opérateurs.

Type	*Calculs possibles*	*Exemple*
int	Opérations entières (la division provoque un arrondi vers le bas)	5/2 = 2
float	Toutes opérations arithmétiques	5/2 = 2.5
str	Opérateur +	"a" + "b" = "ab"

Tu pourrais te dire que qui peut le plus peut le moins. Puisque le type float peut tout faire, pourquoi avoir conservé le type int ? Il faut savoir que les valeurs de type int occupent moins d'espace mémoire dans l'ordinateur, et sont moins coûteuses en temps d'exécution. Dès que tu sais que tu n'auras pas besoin de valeur réelle, il faut privilégier le type int. Par exemple, pour un compteur comme celui de notre exemple, il n'y aurait aucun intérêt à consommer deux fois plus d'espace ou plus en utilisant des valeurs réelles pour les variables de 1 à 10.

Il existe quelques autres types en langage Python, mais les trois que nous venons de découvrir suffisent à notre projet.

Le transtypage

Il te sera parfois utile de pouvoir convertir une valeur d'un type vers un autre. Fort heureusement, Python offre tout ce qu'il faut pour répondre à ce besoin, comme le montre le Tableau 9.4.

Tableau 9.4 : Exemples de transtypage.

Formule	*Résultat*	*Exemple*
int(variable)	Produit une valeur int	int(5.9) = 5
float(variable)	Produit une valeur float	float(5) = 5.0
str(variable)	Produit une chaîne	str(5.9) = "5.9"

Tu constates que lorsque nous écrivons int(5.9), le transtypage fait perdre tout ce qu'il y avait après le point. Il n'y a aucune tentative d'arrondir à 6. N'oublie pas cela si tu comptes faire des calculs avec des valeurs entières.

Combiner deux types dans

l'affichage

Avec ce que nous venons d'apprendre, nous allons pouvoir parvenir à nos fins : utiliser une seule instruction d'affichage pour afficher du texte et la valeur de la variable qui sert de compteur. Nous allons nous servir de la fonction de transtypage `str()`.

Si tu veux tester cet exemple, sauvegarde une troisième version du programme d'essai sous le nom `compteur_3.py` et retouche le code pour ajouter la fonction puis lance l'exécution.

Listing 9.5 : Code source de compteur_3.py

```
for cpt in range(1,11):
    print "Le nombre est : " + str(cpt)
```

Revenons au projet de devinette

Nous pouvons maintenant reformuler les différentes étapes de notre jeu :

1. Nous commençons par une boucle de répétition for pour énumérer de 1 à 10.
2. Nous affichons la valeur actuelle du compteur.
3. Nous demandons au joueur de saisir oui si c'est la valeur à laquelle il pensait ou non dans le cas contraire.

   Nous continuons à lui demander de saisir tant qu'il n'a pas saisi exactement l'une des deux réponses acceptées.

4. Si le joueur a répondu oui, nous affichons le message de victoire. Dans le cas contraire, nous revenons à l'Étape 2 pour passer au tour de boucle suivant.

   Il nous faut prévoir un cas particulier : nous sommes arrivés au chiffre 10 mais le joueur n'a jamais répondu oui. Cette situation est normalement impossible, sauf si le joueur a menti. Nous allons donc ajouter une dernière étape :

5. Si le compteur atteint la valeur 10 alors que le joueur n'a toujours pas répondu oui, afficher un message comme quoi c'est un tricheur.

Tu en sais maintenant assez pour écrire le code complet du projet. Je te propose de chercher un peu par toi-même avant d'étudier le code source suivant.

Pour créer cette troisième version, ouvre une nouvelle fenêtre d'édition que tu enregistres immédiatement sous le nom `devinet_3.py`. Saisis ensuite les lignes suivantes :

Listing 9.6 : Code source de devinet_3.py

```
for nDevi in range(1,11):
    sReponse = raw_input("C'est " +
str(nDevi) + ", oui ou non? ")
    while sReponse != "oui" and sReponse !=
"non":
        sReponse = raw_input("C'est " +
str(nDevi) + ", oui ou non? ")
    if sReponse == "oui":
        print "J'ai lu dans ton cerveau !"
```

```
            break
if sReponse == "non":
    print "Serais-tu un tricheur ?"
```

Lance ensuite l'exécution pour tester ton programme (Figure 9.6).

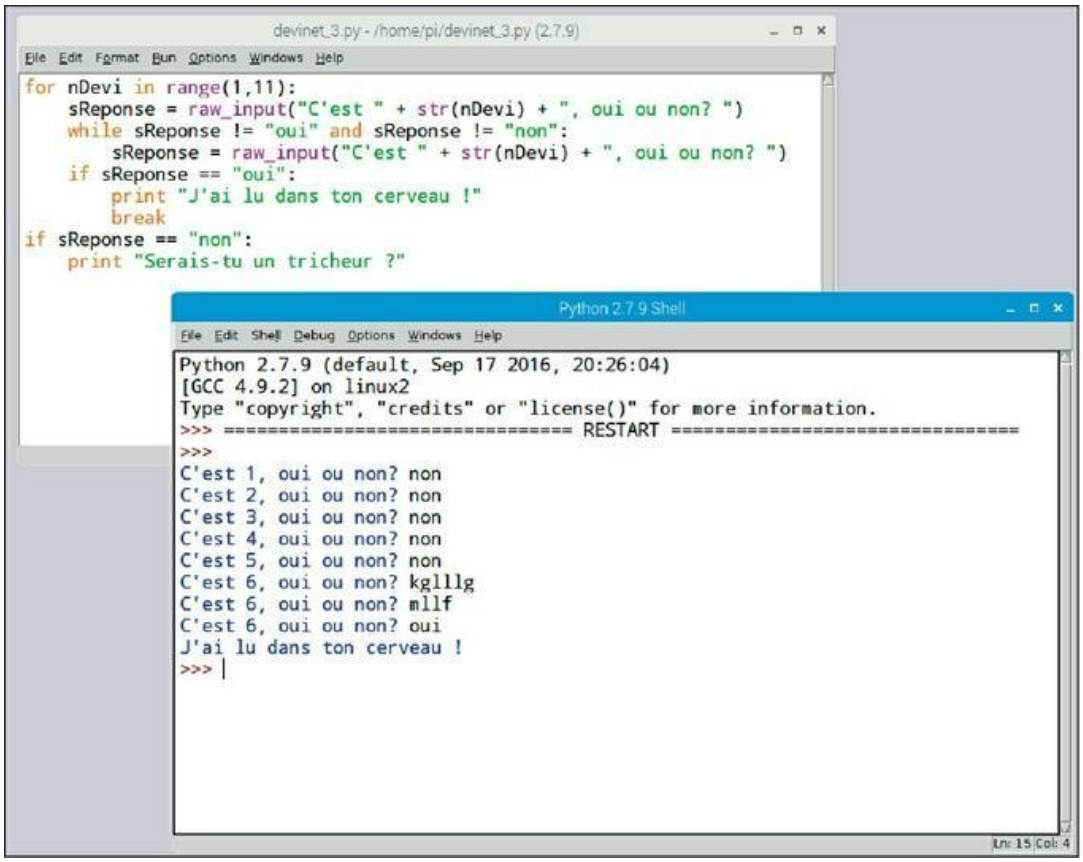


Figure 9.6 : Exemple d'exécution de devinet_3.py.

La largeur des pages de ce livre ne permet peut-être pas d'imprimer toutes les lignes d'instruction du code source sur une seule ligne physique. Dans ta saisie, n'ajoute pas de retour de ligne. Inspire-toi plutôt du contenu de la figure pour voir où se trouvent les vrais sauts de ligne et indentations. Sache que le code source est également disponible sur le site Web de l'éditeur.

La puissance des fonctions

Notre projet fonctionne comme prévu, mais il n'est pas très lisible.

On considère qu'un bon code source est un code qui reste facile à lire et qui ne comporte pas de répétitions inutiles. Nous pouvons donc améliorer celui de notre projet.

Les répétitions d'instructions sont à bannir parce qu'elles rendent difficiles les évolutions du programme. Lorsque tu as besoin de modifier une instruction qui est citée plusieurs fois dans le code source, il faut appliquer la même retouche plusieurs fois, sans en oublier une. Et les oublis sont fréquents.

Un code source mal rédigé risque plus facilement de contenir des erreurs que l'on appelle des bogues. Et plus tu vas faire grossir ton projet, plus tu risques d'en introduire.

Comme quasiment dans tous les langages, Python offre une solution définitive à ce genre de problème. Il suffit de réunir chaque bloc de plusieurs instructions que tu comptes utiliser de façon répétée dans un ensemble indépendant que l'on appelle une *fonction*.

Le terme fonction vient du monde des mathématiques. Une fonction réalise une action sur des données et renvoie un résultat. Le comportement exact de la fonction peut varier en fonction de la valeur ou des valeurs que tu lui transmets au démarrage, c'est-à-dire en fonction de ses arguments d'entrée ou paramètres.

Une fonction se caractérise par quatre éléments :

- un nom unique (son identifiant) ;
- une ou plusieurs variables d'entrée (ses arguments) ;
- une ou plusieurs lignes d'instructions (ses traitements) ;
- une instruction pour renvoyer un résultat.

Les arguments d'entrée ne sont pas obligatoires. Imagine par exemple une fonction dont le seul but est de renvoyer l'heure actuelle. Elle n'a pas besoin d'argument initial, sauf si tu veux lui faire renvoyer une heure différente en précisant un fuseau horaire.

De même, il n'est pas obligatoire de renvoyer une valeur. Dans ce cas, on renvoie la pseudo-valeur None. En revanche, il faut toujours donner un nom unique à la fonction, et indiquer au moins le mot-clé return si elle n'a encore aucun contenu dans sa version initiale.

Les deux termes « argument » et « paramètre » sont utilisés indifféremment pour les valeurs fournies au démarrage de la fonction. Une fonction peut accepter plusieurs arguments d'entrée, séparés par une virgule.

Définir une fonction

Pour créer le corps d'une fonction, c'est-à-dire la définir, il suffit d'utiliser le mot-clé def.

```
def nom_fonction(argument)
    instruction de traitement
    instruction de traitement
    return valeur
```

Utiliser une fonction consiste à écrire un *appel de fonction*. Pour appeler, et donc faire exécuter, une fonction, il suffit d'indiquer son nom ainsi que les noms des variables dont les valeurs doivent être transmises au démarrage :

```
nom_fonction(nom_variable)
```

En général, on récupère dans une variable la valeur que renvoie la fonction :

```
ma_valeur = nom_fonction(nom_variable)
```

Écrire le nom de la fonction de cette façon revient à invoquer ou appeler la fonction. Cela provoque l'exécution de toutes les instructions qui ont été réunies dans le corps de la fonction, donc avant le mot-clé `return`. Tu peux appeler la fonction un nombre de fois illimité. Il n'y a plus aucune répétition des instructions correspondantes dans le code source.

Le corps d'une fonction

Le premier objectif d'une fonction est de constituer un bloc d'instructions réutilisables. Simultanément, cela rend l'ensemble du code source plus lisible.

Si nous revenons à la dernière version de notre projet de devinette, tu constates que les lignes qui demandent de saisir une réponse avec la fonction `raw_input()` sont vraiment assez longues, et peu lisibles. Il y a deux solutions pour résoudre ce problème.

Nous pourrions définir une fonction pour chaque utilisation de la fonction de saisie puis appeler cette fonction.

Une solution plus intéressante consiste à extraire la totalité des lignes d'instructions qui gèrent la demande de saisie et l'analyse de la réponse pour constituer un bloc indépendant qui va devenir une fonction.

Il faut d'abord savoir que la définition d'une fonction doit toujours être présente dans le code source avant le premier appel à cette fonction. Dans d'autres langages, cela n'est pas obligatoire, mais dans Python, c'est ainsi qu'il faut procéder.

1. Si nécessaire, ouvre la fenêtre d'édition sur la dernière version de notre projet (`devinet_3.py`) et enregistre le projet sous un nouveau nom, `devinet_4.py`.

2. Place-toi avant le premier caractère du code source et frappe la touche Entrée pour insérer une ligne vide.

3. Remonte au début de la première ligne et insère tout le corps de définition de la fonction suivante.

Listing 9.7 : Définition d'une fonction dans devinet_4.py.

```
def demander(nEssai):
    sMsg = "C'est " + str(nEssai) + ", o ou
n? "
    sRep = raw_input(sMsg)
    while sRep != "o" and sRep != "n":
        sRep = raw_input(sMsg)
    return sRep
```

Si tu observes bien la deuxième ligne, tu constates que je t'ai proposé une autre technique pour rendre le code plus lisible. Je prépare une chaîne de caractères qui contient la totalité du message que je m'apprête à afficher. Cette chaîne porte le nom `sMsg`. Je réutilise ensuite cette variable deux fois comme argument de `raw_input()`.

Il ne reste plus qu'à simplifier le bloc d'instructions principal du programme, celui qui commence par l'instruction conditionnelle `for`. Voici la nouvelle version de ce bloc principal :

Listing 9.8 : Suite et fin de la nouvelle et dernière version du projet devinet_4.py

```
for nDevi in range(1,11):
    sReponse = demander(nDevi)
    if sReponse == "o":
        print "J'ai lu dans ton cerveau !"

        break
```

```
if sReponse == "n":
    print "Serais-tu un tricheur ?"
```

Quelques remarques concernant cette nouvelle variante :

- la première ligne ne change pas ;
- la deuxième ligne est devenue beaucoup plus simple puisque nous effectuons un appel à la nouvelle fonction `demander()` ;
- le premier test de la valeur saisie a été simplifié. Nous ne demandons maintenant que de frapper la lettre o ou la lettre n, ce qui est plus pratique pour faire des tests ;
- globalement, tout le bloc while a été renvoyé dans la nouvelle fonction.

Tu seras d'accord sur le fait que le programme est devenu beaucoup plus lisible. On dit qu'il a été restructuré. Tu remarques également que la fonction utilise un verbe comme nom. C'est assez logique, puisqu'une fonction effectue une action.

Cette structuration correspond bien à l'approche suggérée en début de chapitre : décomposer tout problème en problèmes plus simples.

N'hésite pas à tester la nouvelle version du programme. Tu dois obtenir quelque chose de similaire au contenu de la Figure 9.7.

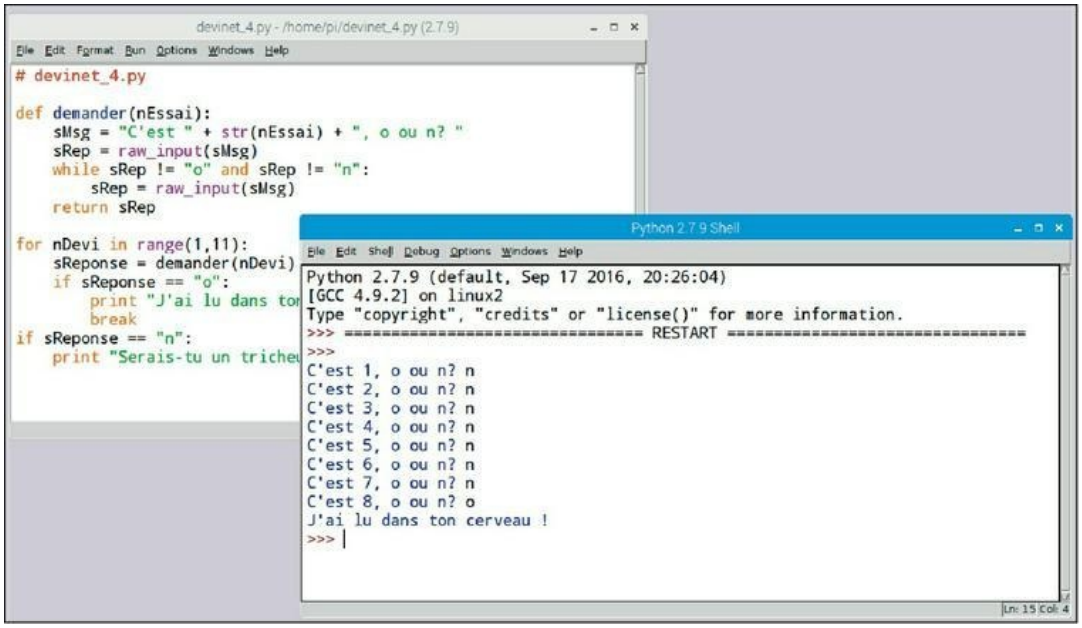


Figure 9.7 : Exécution de devinet_4.py.

Dans la fonction, tu constates que j'ai utilisé de nouvelles variables `sMsg` et `sRep`. Ce sont des variables locales, c'est-à-dire qu'elles ne sont pas utilisables en dehors de la fonction. En général, cette manière de faire est une bonne pratique, car il évite de modifier une variable par mégarde.

Le seul moyen pour connaître la valeur d'une variable locale en dehors de la fonction consiste à la faire renvoyer en indiquant son nom à la suite du mot-clé `return`, comme nous le faisons ici.

Une variante magique

Il y a une règle secrète dans le monde de la programmation que personne ne t'avouera : un bon code source va te paraître plus intelligent que toi-même, alors qu'un mauvais code source te paraître plus bête.

Il en va un peu comme avec les prestidigitateurs. Tant que tu ne connais pas le truc, tu restes impressionné, mais une fois qu'il t'a été dévoilé, tu es souvent déçu. Mais lorsque la technique est vraiment futée, cela ressemble à de la vraie magie.

Dans sa version actuelle, notre jeu n'est vraiment pas intelligent. Comment pourrait-on l'améliorer ? Il nous faut chercher une manière plus efficace de deviner le nombre au lieu de passer en revue toutes les valeurs en partant de la plus petite.

As-tu une idée de la technique que nous pourrions appliquer ? On pourrait bien sûr chercher le nombre au hasard, comme en lançant un dé à dix faces. Cela prendrait autant de temps en moyenne et il faudrait stocker quelque part tous les coups déjà joués.

Les programmeurs adorent passer du temps à chercher des techniques plus efficaces pour réaliser des opérations très courantes comme le tri dans une série de données ou la recherche parmi des valeurs numériques ou des textes. En réfléchissant bien au problème, tu peux trouver tout seul un bon algorithme. La plupart des gens vont puiser la réponse dans des livres ou sur Internet.

Dans le cas qui nous intéresse, un algorithme efficace s'appelle la *recherche binaire*. Malgré son nom, c'est une technique très simple. Pour le premier essai, tu choisis la valeur qui est exactement au milieu de la plage et tu demandes au joueur si son nombre est supérieur ou pas.

En fonction de sa réponse, tu n'as plus que la moitié de la plage de valeurs à gérer. Il suffit de répéter cette opération sur le quart, puis le huitième de la plage, et ainsi de suite.

Tu finis par ne plus avoir que soit deux, soit trois nombres parmi lesquels choisir. Lorsqu'il n'y en a plus que deux, tu demandes si c'est le plus grand des deux. Si c'est le cas, tu as trouvé. Si ce n'est pas le cas, c'est l'autre nombre.

Si tu aboutis à trois nombres, tu commences par chercher celui du milieu. Si la valeur est trop faible, tu sais que c'est le nombre le plus grand des trois. Dans le cas contraire, tu n'as plus que deux nombres, et tu appliques la recette du paragraphe précédent.

Grâce à cette technique, lorsque tu dois deviner un nombre entre 1 et 10, tu parviens toujours au résultat en trois ou quatre coups, comme le montre la Figure 9.8.

Figure 9.8 : Exemple de recherche binaire.

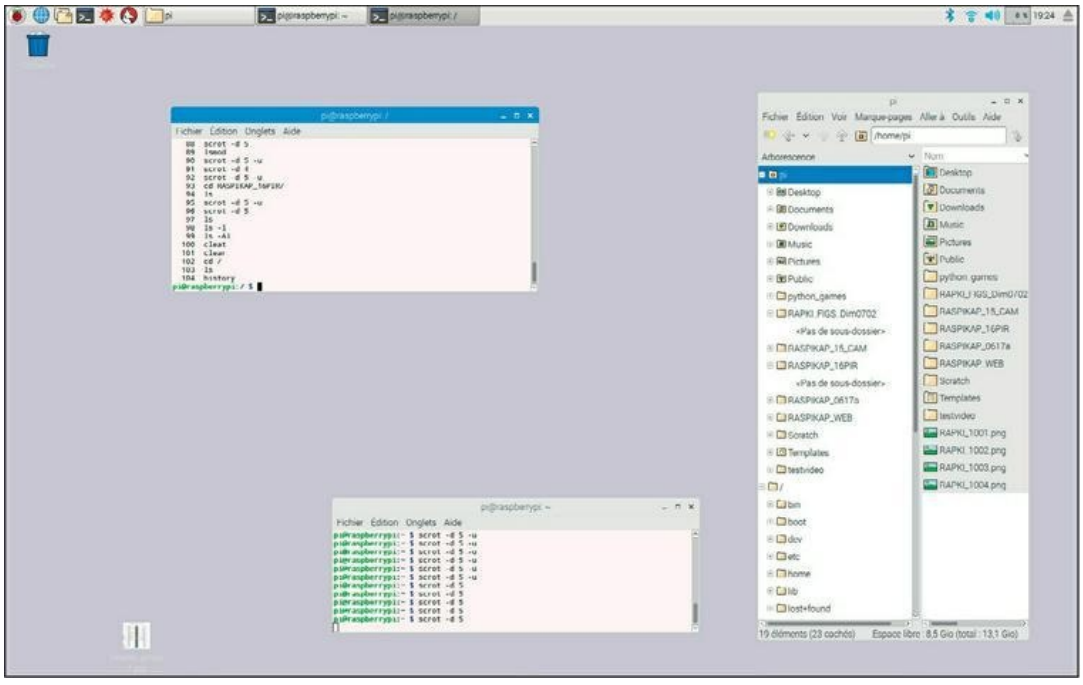
Saurais-tu écrire seul le code source correspondant à l'utilisation de cette technique de recherche binaire ? Je fournis dans les exemples du livre (disponibles sur le site de l'éditeur) la réponse à cet exercice. Mais avant de dévoiler ce tour de magie, essaie par toi-même. Comme d'habitude, il faut toujours décomposer le problème en problèmes plus simples.

La recherche binaire montre encore plus son intérêt lorsqu'il faut deviner le nombre dans une plage plus vaste. Modifier par exemple le projet pour trouver un nombre entre 1 et 1000. Combien faut-il d'essais selon toi ? Fais des tests ; tu verras qu'il faut beaucoup moins de 1000 essais, même moins de 100.

Chapitre 10
Les commandes du système Linux

Le système Linux ressemble à un iceberg, non pas parce qu'il serait gros, froid et risquerait de faire sombrer des navires, mais tout simplement parce que le bureau graphique auquel tu es habitué n'est que la partie émergée du système.

Pour pouvoir tirer profit de toute la puissance de ce système Linux, il faut apprendre à dompter les commandes en mode texte et apprendre à naviguer dans l'arborescence de son système de fichiers.



La ligne de commande

Dans les années 1950 à 1980, soit pendant la préhistoire de l'informatique, on ne pouvait pas utiliser de bureau graphique. L'affichage graphique n'existait pas, ni la souris, du moins la souris informatique.

Le seul moyen d'interagir avec un ordinateur consistait à saisir des commandes au clavier. Lorsque tu voulais afficher les noms de tes fichiers, il fallait saisir une commande. Pour démarrer ton logiciel préféré, il fallait une autre commande. Lorsque tu voulais arrêter l'ordinateur afin que lui comme toi puissent se reposer un peu, il fallait encore une autre commande.

Avec les ordinateurs les plus récents, cet ancien système reste disponible, même s'il est caché. Si personne ne t'en informe, tu ne peux pas savoir qu'il est disponible.

Cette manière de contrôler un ordinateur correspond au mode ligne de commande. Il permet de faire bien plus de choses qu'à partir du bureau graphique et de la souris.

Tu trouveras aisément sur le Web des photographies des premiers ordinateurs qui fonctionnaient en mode texte. Cherche par exemple le terme « terminal » ou « MS-DOS » . À cette époque, on ne pouvait afficher que du texte, aucune photo, vidéo et encore moins des graphiques de jeux vidéo. Les imprimantes étaient lourdes et bruyantes et utilisaient du papier à bandes perforées. De nos jours, les ordinateurs sont devenus bien plus silencieux, et tu peux dorénavant en avoir un dans ta poche sous forme d'un telliphone.

L'exactitude du mode texte

Le mode texte ne pardonne aucune faute de frappe. Tu ne peux pas confondre une majuscule avec une minuscule, car le résultat sera différent. Dans chaque ligne de commande que tu vas saisir, il te faudra être parfaitement exact, car sinon, soit il ne se passera rien, soit tu vas provoquer une catastrophe.

Par définition, un ordinateur ne prend aucune initiative. Si tu lui demandes de faire une erreur, il va obéir aussi bien que si tu lui demandes un traitement sensé. Dans le meilleur des cas, il ne va pas

comprendre ce que tu demandes et répondre par un message d'erreur plus ou moins compréhensible.

Cette contrainte d'exactitude pourrait te faire croire que le mode texte représente une manière compliquée de faire des choses simples. Dans l'interface graphique, il est facile d'afficher une liste de fichiers et de dossiers. Sur la ligne de commande, c'est moins simple. Il en va de même pour créer un fichier et quasiment pour toutes les autres opérations courantes que l'on a besoin de faire avec un ordinateur.

Mais si le mode texte existe toujours, c'est qu'il offre des avantages incomparables. Une fois que tu as appris à le maîtriser, tu sauras enchaîner plusieurs commandes pour réaliser des opérations complexes. C'est par exemple le seul moyen de changer facilement le nom de tous les fichiers d'un dossier sans devoir les ressaisir un par un. C'est également en mode texte que tu peux retrouver facilement tous les fichiers d'un certain type, par exemple toutes les photographies ou tous les fichiers audio MP3, pour les copier d'un geste vers un seul dossier destinataire, afin de les rassembler. Tu peux même programmer des commandes pour les faire déclencher automatiquement une fois par heure, par jour ou par semaine.

La ligne de commande n'est pas une manière complexe de faire des choses simples. C'est une technique qui permet de réunir des commandes élémentaires à la suite les unes des autres pour obtenir des traitements puissants.

La ligne de commande permet d'adapter ton ordinateur à tes besoins pour qu'il te rende plus de services et qu'il les réalise plus rapidement.

L'invite de commande en pratique

Les versions actuelles du système d'exploitation de ton Raspberry Pi démarrent directement en mode graphique et affichent le Bureau. Tu peux cependant à tout moment revenir au mode texte des origines en utilisant une fenêtre de terminal.

Pour faire apparaître cette fameuse fenêtre, ouvre le menu principal du bureau et choisis `LX Terminal` dans les Accessoires.

Tu vois apparaître la fenêtre de l'application LX Terminal. Seule la première ligne contient du texte. Le contenu de la fenêtre ressemble à ce que tu verrais si tu disposais d'un écran d'affichage ne sachant travailler qu'en mode texte.

Le texte affiché incarne l'*invite de commande* du système qui se termine par le symbole dollar. Ce que tu peux saisir va apparaître à droite de ce symbole.

La Figure 10.1 montre l'aspect initial du terminal.

Pour plus de confort, j'ai changé les couleurs de fond et de texte de la fenêtre *via* Édition/Préférences/Apparence.

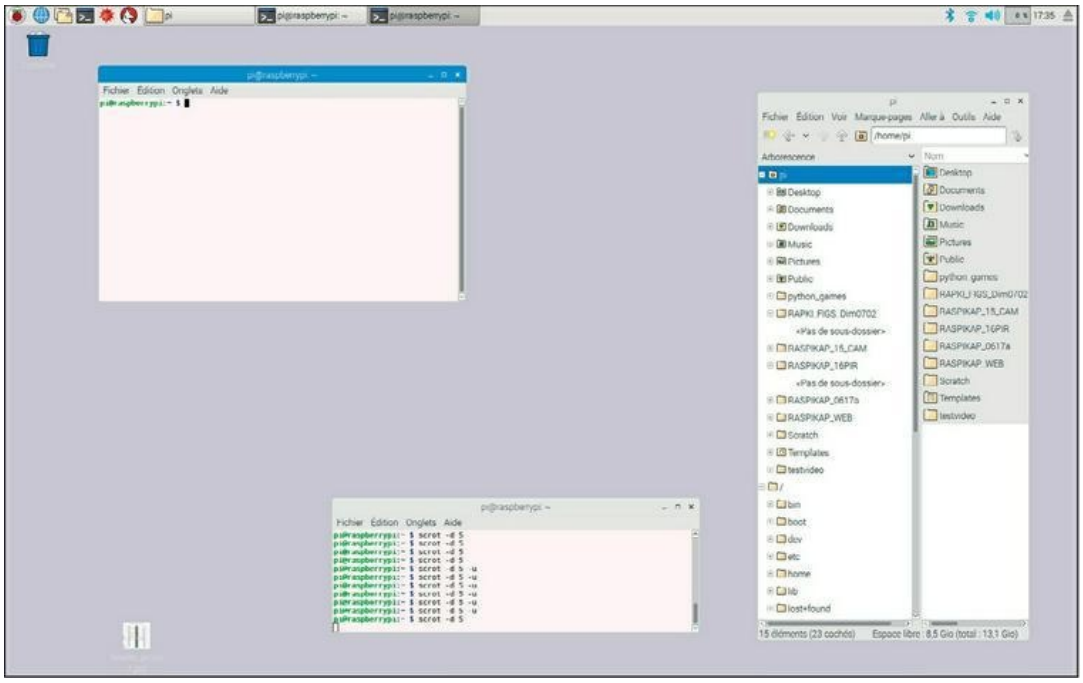


Figure 10.1 : La fenêtre du terminal dans son état initial.

Lorsque cette fenêtre est la fenêtre active, tu peux directement saisir une commande à l'endroit où se trouve le curseur de saisie sur la ligne de l'invite.

Le texte de l'invite permet d'abord de savoir que le système est prêt à exécuter la prochaine commande que tu vas saisir. Il en profite pour rappeler le nom de l'utilisateur et celui de l'ordinateur. L'invite contient également un rappel du nom du dossier ou répertoire dans lequel tu te trouves actuellement.

Dans ce chapitre, j'ai volontairement augmenté la taille des caractères affichés dans la fenêtre de terminal afin de faciliter la lecture du livre. Dans tes exercices pratiques sur ton ordinateur, les lettres seront sans doute affichées dans une taille plus faible. Tu peux à tout moment modifier la taille des caractères en ouvrant le menu `Édition` et en choisissant `Préférences`.

De drôles de noms de commandes

Le système Linux est livré avec des centaines de commandes, au point que personne ne se souvient de toutes et de toutes leurs options.

Quel que soit ton niveau d'expérience en informatique, tu ne peux pas deviner à quoi sert chaque commande uniquement avec son nom, car c'est en général une abréviation ou un acronyme.

Certains croient même que les commandes en mode texte sont des formules d'incantation magique. Il faut connaître ces formules pour réussir des exploits.

Puisque tu ne peux pas deviner à quoi sert chaque commande, il te faut apprendre à quoi correspond le nom de chacune d'elles, en cherchant sur le Web ou en te faisant aider. Prépare-toi à passer un certain temps au départ pour mémoriser celles dont tu auras le plus besoin.

Inutile de chercher l'expression « commande magique » sur le Web. Ces commandes n'ont en réalité rien de magique. Cette aura de magie se dissipera au fur et à mesure que tu prendras possession d'un nombre de plus en plus grand de commandes.

Les options de ligne de commande

Quasiment toutes les commandes acceptent des options qui influent sur leur fonctionnement.

En général, les options commencent toujours par un tiret, lui-même suivi d'une ou plusieurs lettres ou chiffres. Il y a quelques options qui commencent par deux tirets consécutifs, mais elles sont rares.

La Figure 10.2 montre trois exemples d'utilisation de la commande de listage ls, d'abord sans option, puis avec deux options successives. Cette commande sert à afficher les noms des fichiers trouvés dans le répertoire courant.

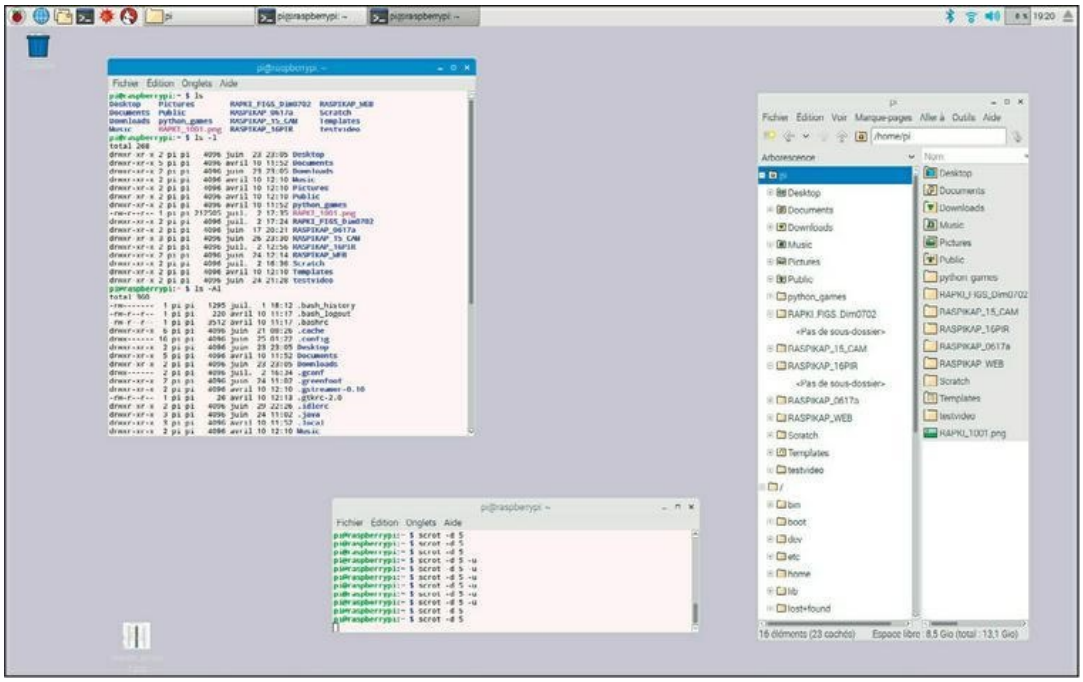


Figure 10.2 : Exemple d'utilisation de la commande ls.

Parmi les nombreuses options des commandes, seul un petit groupe est utile au quotidien. Les autres options servent à répondre à des situations rares, mais elles ont le mérite d'exister.

Par exemple, pour la commande de listage ls, le fait d'ajouter l'option -l en écrivant la commande ls -l puis en frappant la touche Entrée permet d'afficher pour chaque fichier en plus du nom, la taille, la date de création et le nom du créateur.

En ajoutant l'option -a, tu fais aussi afficher la liste des noms des fichiers normalement invisibles.

Certaines commandes permettent de combiner les lettres de plusieurs options. Pour afficher tous les fichiers cachés avec tous leurs détails, nous pouvons utiliser la commande qui s'écrit de la façon suivante :

```
ls -Al
```

Comment ? Mon système d'exploitation me cache des fichiers ? En fait, ces fichiers ne sont pas vraiment invisibles. Ils sont simplement masqués de l'affichage normal, car ils n'ont pas d'intérêt au quotidien. Ce sont surtout des fichiers de configuration des applications, de contrôle de l'ouverture de session et autres considérations techniques dont tu n'as généralement pas besoin de te soucier. En les cachant, cela évite d'encombrer l'affichage. Autrement dit, il n'est pas inutile de ne les montrer que sur demande.

Quelques commandes à apprendre

Puisque tu ne peux pas deviner les noms des commandes et de leurs options, il faut savoir où les trouver. Celles que tu utiliseras le plus fréquemment finiront par s'imprimer dans ta mémoire.

Une technique de mémorisation souvent utilisée consiste à se concocter une petite fiche avec les noms des commandes et les options que l'on a utilisées au moins une fois. À moins d'avoir une mémoire d'éléphant, ce genre de fiche te sera très utile, car il arrivera nécessairement un jour où tu te demanderas comment s'écrit la fameuse commande que tu avais utilisée quelques mois auparavant pour faire une opération dont tu as à nouveau besoin.

Heureusement, grâce à Internet, tu peux facilement obtenir une référence de toutes les commandes Linux avec leurs options, du moins les plus utilisées. La Figure 10.3 montre par exemple la page principale du site d'information `wiki.linux-france.org`.

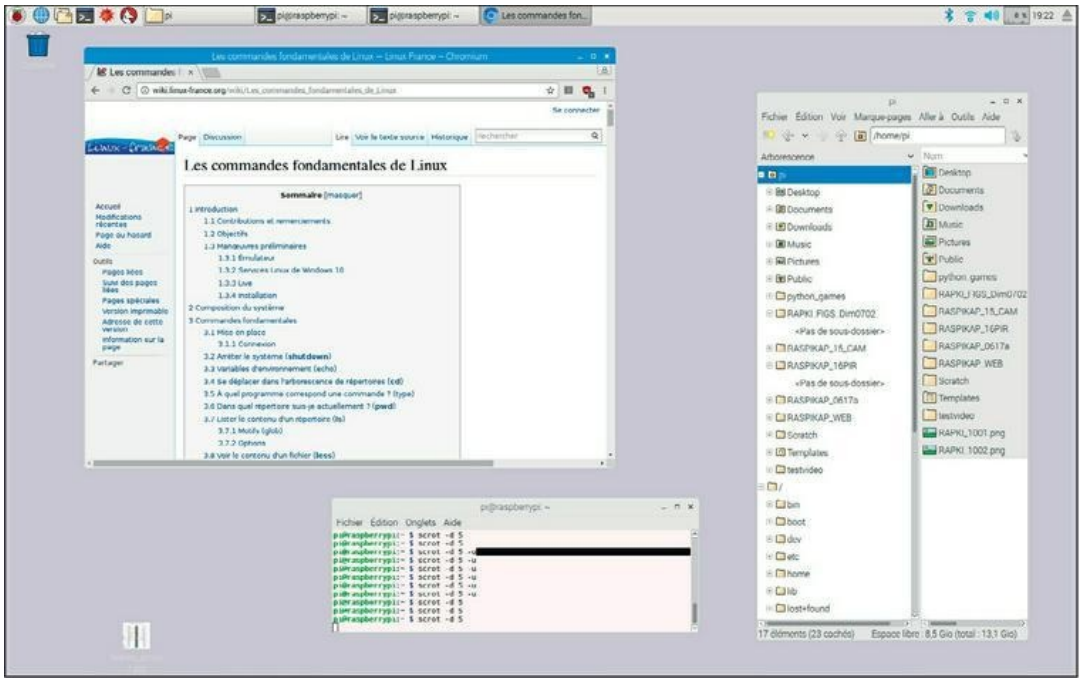


Figure 10.3 : Exemple de référence des commandes Linux.

Il existe sous Linux, comme sous Unix, une commande portant le nom man (qui vient de manuel). Il suffit de saisir cette commande suivie du nom de la commande pour laquelle tu désires des informations pour voir s'afficher une sorte de système d'aide. Hélas, le résultat est entièrement en anglais et il n'est pas facile d'accès. De plus, les options sont fournies dans l'ordre alphabétique et non dans l'ordre décroissant de leur taux d'utilisation. Tu peux tout à fait voir apparaître en premier une option que personne n'utilise. Mieux vaut utiliser les références sur le Web comme celle indiquée un peu plus haut.

Les deux commandes cd et ls

Commençons par deux commandes très utilisées et totalement inoffensives.

La commande `ls`, un nom abrégé pour liste, sert à afficher les noms des fichiers qui se trouvent dans le répertoire courant. C'est l'équivalent de l'affichage du contenu d'un dossier dans le Gestionnaire de fichiers de l'interface graphique.

La commande `cd` est l'abréviation de Change Directory. Elle sert à changer de répertoire courant. C'est grâce à elle que tu peux naviguer parmi les différents répertoires ou dossier du système de fichiers de Linux. Pour passer dans un autre répertoire, tu commences par indiquer le nom de la commande, une espace puis le chemin d'accès au nouveau répertoire. Le répertoire courant s'appelle également répertoire de travail (*Working Directory*). Une fois dans ce répertoire, tu peux afficher le contenu, copier ou supprimer un fichier et faire d'autres opérations sans avoir besoin de préciser à nouveau le nom du répertoire.

Tu as sans doute remarqué que j'utilisais le terme *répertoire* à la place du terme *dossier*. Ils sont absolument synonymes, sauf que les interfaces graphiques ont choisi le terme anglais *folder* pour parler des répertoires, et *folder* se traduit en dossier. Au lieu de chercher à savoir si faut utiliser un terme plutôt que l'autre, tiens-toi à la règle suivante : dans l'interface graphique, on appelle cela un dossier, et dans l'interface en mode texte, on appelle cela un répertoire. Dans les deux cas, il s'agit d'un conteneur qui peut recevoir des fichiers ou des sous-dossiers ou sous-répertoires.

Puisque nous avons déjà vu comment utiliser `ls` dans le répertoire courant, voyons comment afficher le contenu d'un autre répertoire du système.

Commence par taper la commande `cd /`puis valide par la touche Entrée. Cela te fait entrer dans le répertoire racine de tous les répertoires du système. C'est dans ce répertoire que se trouvent tous les sous-répertoires contenant tous les fichiers. Saisis la commande `ls` puis valide par Entrée. Tu dois voir apparaître la liste des sous-répertoires de premier niveau (Figure 10.4).

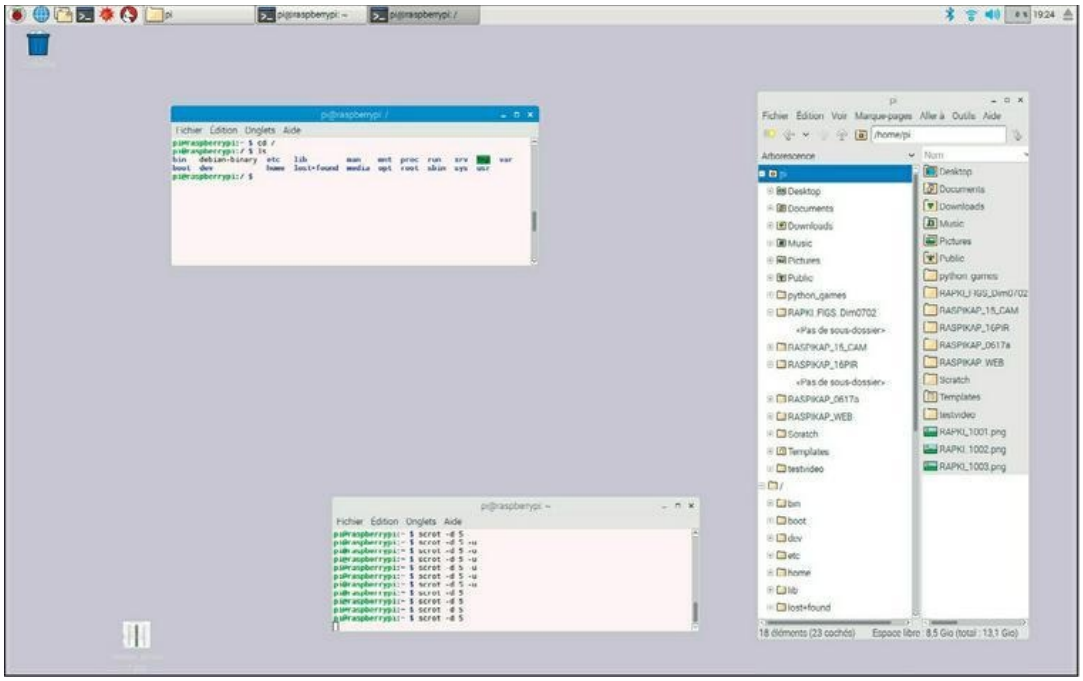


Figure 10.4 : Liste des sous-répertoires de premier niveau.

Dans le Chapitre 5, nous avons appris que les fichiers apparaissaient dans une structure en forme d'arbre. Le tronc de l'arbre correspond au répertoire racine, qui est symbolisé par la barre oblique isolée, /. Il contient les sous-répertoires de premier niveau, qui en contiennent d'autres. Cette structure arborescente est utilisée aussi bien sous Windows que sous macOS, car ces systèmes ont hérité des premiers systèmes en mode texte de type Unix et donc Linux.

Comme dans un arbre, chaque fichier qui constitue une feuille est accessible en suivant un chemin depuis la racine, c'est son chemin d'accès. (Retourne lire le Chapitre 5 si nécessaire).

Pour passer dans un autre sous-répertoire, tu utilises la commande cd en la faisant suivre du chemin d'accès qui mène à ce sous-répertoire. À la différence du Gestionnaire de fichiers, tu ne disposes pas d'une aide visuelle pour te repérer. C'est dans ta tête que tu dois te construire la représentation de l'arborescence pour pouvoir passer facilement d'une branche à une autre.

Pour revenir à ton répertoire personnel principal, si tu as utilisé le compte utilisateur initial, utilise la commande suivante :

```
cd /home/pi
```

Je rappelle que pour exécuter une commande en mode texte, il faut frapper la touche Entrée. J'imagine que tu l'as déjà compris. Dans le mode interactif cette validation de la commande est obligatoire. Elle ne l'est pas lorsque tu réunis les commandes dans un script, mais c'est une autre histoire.

Ne rien céder au confort

Lorsque tu dois naviguer en mode texte d'un répertoire vers un autre, la saisie du chemin d'accès complet peut parfois devenir complexe. Pour te donner un peu plus de confort, la commande cd accepte un certain nombre de raccourcis, comme le montre le Tableau 10.1.

Tableau 10.1 : Raccourcis de la commande cd.

Commande	*Résultat*
cd ~	Retour au répertoire personnel.
cd /	Accès au répertoire racine.
cd nom	Descend dans le sous-répertoire du répertoire courant tel qu'il est mentionné.
cd ..	Remonte d'un niveau dans l'arborescence jusqu'à atteindre le répertoire racine. Ne pas oublier l'espace avant les ..
cd /chemin	Entre directement dans le répertoire mentionné, le chemin d'accès est absolu car il commence depuis la racine grâce au symbole /.

Pour afficher le contenu d'un répertoire, tu peux directement indiquer le chemin d'accès à la suite de la commande ls. Cela t'évite de devoir te rendre dans ce répertoire d'abord avec cd. La commande cd reste néanmoins utile pour toutes les opérations de copie, de déplacement ou de changement de nom. Mieux vaut se trouver dans le répertoire pour vérifier le résultat de ce que l'on a fait. Nous en reparlerons dans le chapitre suivant.

Fais maintenant quelques essais d'utilisation de la commande cd avec les différents raccourcis proposés dans le Tableau 10.1. Tu constates que le nom du répertoire courant est toujours mentionné dans l'invite de commande. Ainsi, tu sais toujours où tu te trouves dans l'arborescence.

Pour éviter que l'invite de commande prenne trop de place dans la ligne, les répertoires les plus utilisés sont symbolisés par un raccourci. Pour ton répertoire personnel, c'est le signe ~. Pour revoir le chemin d'accès complet d'un tel répertoire, tu peux utiliser la commande pwd sans aucune option. Il s'agit d'un sigle pour *Print Working Directory*, c'est-à-dire a*ffichage du répertoire de travail*. Bien sûr, cela n'imprime rien vers une quelconque imprimante.

Les principaux répertoires du système Linux

Une fois que tu sais utiliser les deux commandes cd et ls, tu peux partir dans l'exploration de ton système de fichiers. Parmi les lieux que tu vas ainsi visiter, certains sont des salles techniques dans lesquelles tu ne feras que passer, car l'humain n'a rien à y faire.

Les principaux répertoires du premier et du deuxième niveau par rapport à la racine sont montrés dans le Tableau 10.2. Tu constates que chaque chose est à sa place dans le système Linux.

Ne cherche pas à mémoriser tous ces noms. L'essentiel est d'avoir une idée grossière du contenu des répertoires. Lorsque tu seras plus

intime avec ta machine, tu iras de temps en temps réaliser une opération dans certains de ces répertoires.

Tableau 10.2 : Principaux répertoires de Linux.

Répertoire	*Contenu*
/bin	Programmes standard du système Linux.
/sbin	idem pour le super-utilisateur.
/etc	Fichiers de configuration et d'initialisation.
/var	Données courantes, et notamment les fichiers journaux.
/var/log	Fichiers journaux des applications, ce qui permet de rechercher les causes d'une erreur.
/var/www	Répertoire créé lors de la première mise en place d'un serveur Web sur la machine. Il contient les fichiers des pages Web.
/home	Racine relative de tous les comptes utilisateurs de la machine. Au départ, il n'y a que l'utilisateur pi.
/home/pi	Répertoire personnel de l'utilisateur pi.
/root	Normalement vide. Contient les fichiers qui servent au super-utilisateur.
/usr	Fichiers utilisés par tous les utilisateurs.
/usr/bin	Applications accessibles à tous les utilisateurs.
/usr/sbin	Applications réservées au super-utilisateur.

/usr/local	Applications et données qui ne sont disponibles que sur cette machine, et pas sur d'autres machines Linux.
/dev	Fichiers système des périphériques (*devices*). Ne pas y toucher.
/lib	Librairies de fonctions et compléments des applications.
/proc	Informations au sujet des processus en cours d'exécution.
/sys	Fichier du système en cours d'exécution. Ne rien toucher ici.

sudo pour devenir super-utilisateur

Certains de ces répertoires sont inaccessibles avec la commande cd parce qu'ils sont réservés au système.

L'interdiction n'est pas le résultat d'une erreur de frappe. Linux est très pointilleux quant à sa sécurité.

Quand tu ouvres une session, tu travailles avec un compte utilisateur ordinaire. Linux ne fait pas totalement confiance à un tel utilisateur. Voilà pourquoi il t'empêche d'entrer dans certains répertoires.

Comme déjà vu dans le Chapitre 5, il n'y a qu'un moyen pour obtenir tous les droits : devenir super-utilisateur, ce qui se dit également utilisateur racine ou *root*.

Sur la ligne de commande, tu peux accéder à ces super-pouvoirs au moyen de la commande sudo. Il suffit d'ajouter cette commande avant le nom de la commande que tu veux faire exécuter avec tous les

pouvoirs. Linux accepte de te considérer comme super-utilisateur uniquement pour exécuter cette commande.

Tu peux normalement utiliser la commande sudo autant de fois que nécessaire.

Dans certaines variantes de Linux, la commande sudo est collante : tu peux continuer à avoir les droits du super-utilisateur pendant plusieurs minutes. Tu retombes ensuite dans l'état d'utilisateur normal. Ce n'est pas le cas sur le Raspberry. Il faut saisir la commande sudo avant chaque commande pour laquelle tu en as besoin. Si tu veux vraiment devenir super-utilisateur pendant plus longtemps, tu peux utiliser la commande sudo su. Tu sais que tu es passé dans le mode surpuissant, et plus dangereux, parce que l'invite change de couleur. Tu restes alors dans ce mode jusqu'à te déconnecter ou sortir de la fenêtre avec la commande exit.

Un utilisateur Linux sérieux ne travaille jamais en permanence dans le mode super-utilisateur. En effet, une petite erreur peut avoir dans ce mode d'énormes conséquences. Cela a beaucoup moins d'importance dans un ordinateur d'apprentissage tel que le Raspberry, surtout si tu es le seul utilisateur. Te voilà prévenu : lorsque tu passes en mode super-utilisateur, reste sur tes gardes, mais n'hésite pas pour autant à le découvrir.

Les aides à la saisie

Sur la ligne de commande, une seule petite faute de frappe peut entraîner le refus d'exécuter la commande. Il serait sans doute pratique de pouvoir éviter de ressaisir la totalité d'une commande ?

C'est prévu. Il y a plusieurs raccourcis pour t'aider.

Rappel des commandes précédentes

Il suffit d'utiliser la flèche Haut du clavier pour faire afficher la dernière commande saisie. Si tu continues de frapper la même touche, tu passeras en revue tout l'historique des commandes déjà réalisées depuis le début.

Pour lancer la commande que tu as affichée, il suffit de valider directement par la touche Entrée. Linux s'exécute comme si tu venais de la ressaisir.

C'est un raccourci indispensable et c'est le premier à connaître.

Recherche d'une commande

Maintiens les deux touches `Ctrl` et `R` enfoncées puis saisis le début d'une commande. Le système va chercher toutes les commandes déjà utilisées qui correspondent.

Cette option est moins intéressante que la précédente parce que la réponse n'est parfois pas celle recherchée.

Historique de commande

Tu disposes de la commande history qui affiche les précédentes commandes comme le montre la Figure 10.5.

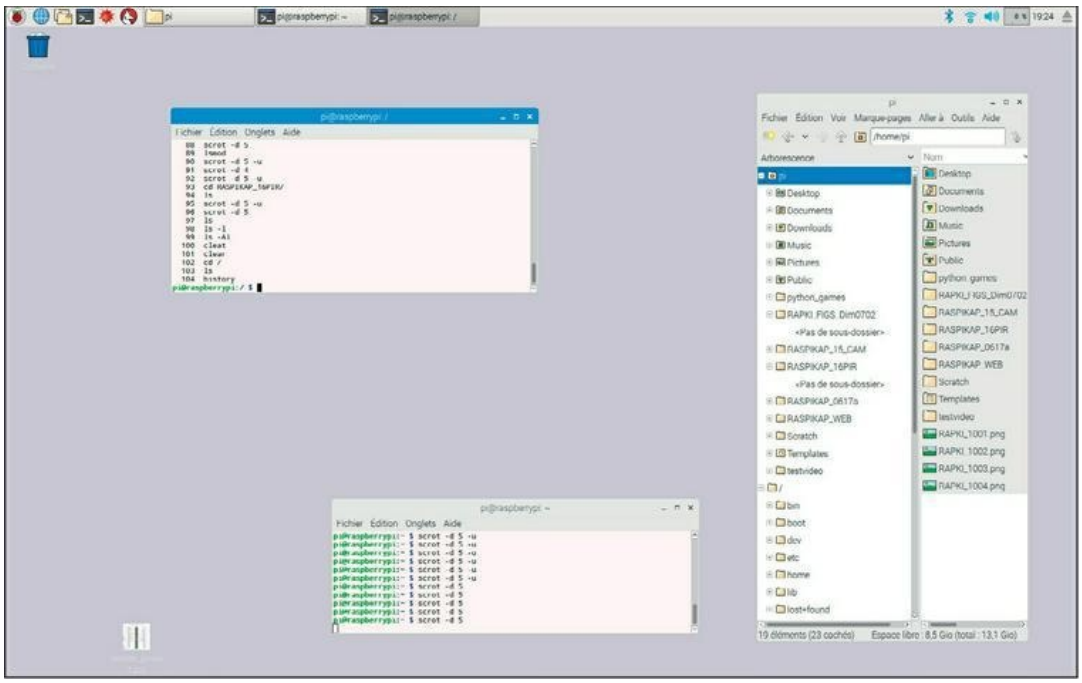


Figure 10.5 : Exemple d'affichage de la commande history.

Quand tu auras utilisé le système pendant un certain temps, la liste de l'historique va commencer à s'allonger. Pour réduire l'affichage, il suffit d'indiquer à la suite du nom de la commande une quantité pour n'afficher que les X dernières commandes. Par exemple, en écrivant history 4, tu n'affiches que les quatre plus récentes commandes.

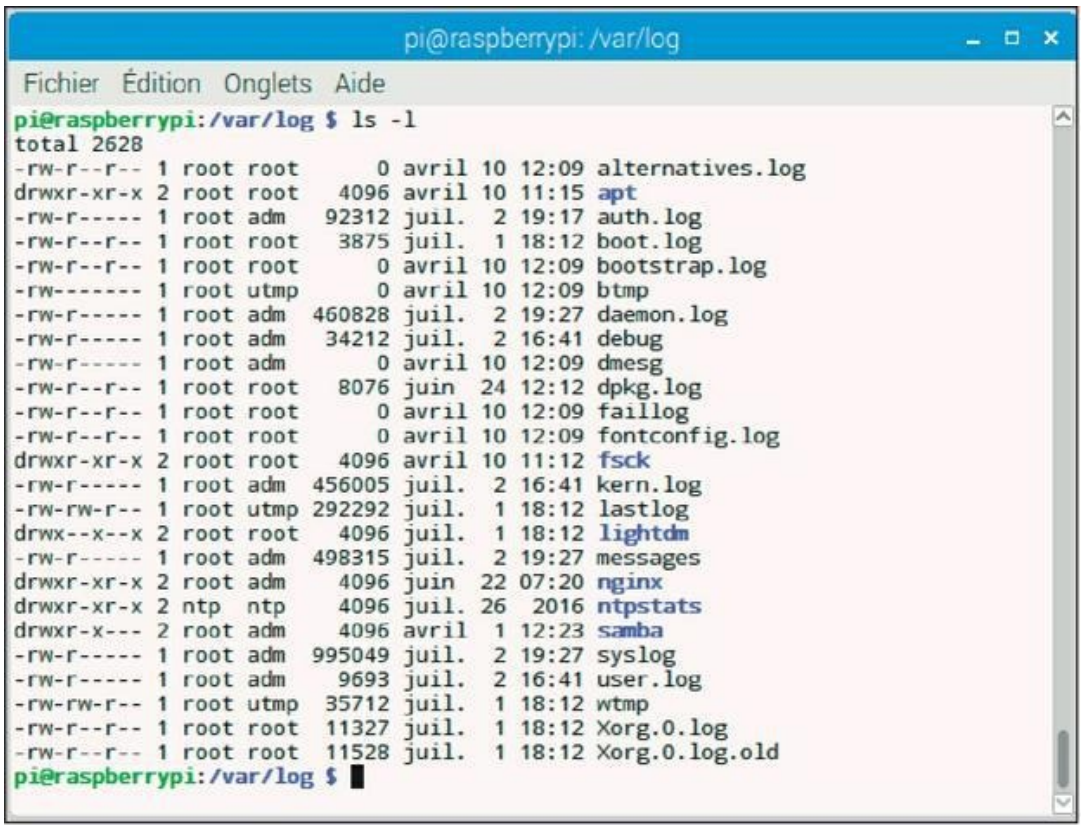
Tu constates par ailleurs que la commande history présente une liste numérotée. Tu peux directement relancer une des commandes en tapant le point d'exclamation suivi immédiatement du numéro. Par exemple, si tu tapes ! 12, tu relances la 12e commande la liste. Cette technique permet d'éviter beaucoup de travail de saisie et d'erreurs, surtout lorsqu'il s'agit d'une ligne de commande longue.

Chapitre 11
Gérer et personnaliser le système

Quelle que soit la version de Linux, y compris Raspian, qui est celle spécifique au Raspberry, ce système mérite plus d'attention que Windows ou macOS. Au départ, tu auras l'impression que l'installation d'une application, la configuration du système, et même la création ou le changement de nom d'un fichier sont moins simples sous Linux.

En revanche, Linux permet aisément d'automatiser certaines des opérations de gestion en créant des scripts. Tu peux même programmer ces scripts pour qu'ils soient déclenchés automatiquement à des moments prédéterminés.

Dans tous les cas, si tu veux devenir un maître d'un système tel que Linux, il faut d'abord en savoir plus sur les fichiers et les répertoires.



Les droits d'accès aux fichiers

Si tu as l'habitude d'utiliser Windows ou macOS, les droits d'accès de Linux vont te paraître compliqués. Cela ne fonctionne pas tout seul. Il faut souvent en tenir compte, même pour une opération aussi simple que la copie d'un fichier ou d'un dossier.

Il n'y a pas moyen d'y échapper : il faut prendre le temps de comprendre comment sont gérés et protégés les fichiers pour pouvoir se servir utilement du système Linux.

Lecture, écriture et exécution

Sous Linux, qu'il s'agisse d'un fichier ou d'un dossier, que l'on appelle répertoire ici, il y a trois choses possibles : lire, modifier, c'est-à-dire écrire, et exécuter.

Les droits d'accès ou permissions portent les noms anglais *Read*, *Write* et *Execute*. Chaque fichier est géré par ces trois droits d'accès. C'est comme cela que tu peux faire en sorte qu'un fichier ne puisse être que lu, en désactivant ses deux droits en écriture et en exécution. À partir de ce moment, tu ne peux plus le modifier, et tu ne peux plus en lancer l'exécution si le fichier est celui d'une application.

À quoi cela sert-il de rendre un fichier accessible seulement en lecture ? Par mesure de sécurité. Un fichier est ainsi protégé contre toute modification.

L'expression « exécution » n'a aucun rapport avec le fait de détruire le fichier. Cela signifie que le contenu du fichier va être considéré comme du code binaire que l'ordinateur peut utiliser pour fonctionner. Ce droit aurait pu s'appeler Run au lieu de Execute. Dans les symboles des droits d'accès, ce droit correspond à la lettre X.

Ce triplet de droits existe en trois exemplaires pour chaque élément, qu'il s'agisse d'un fichier ou d'un dossier :

- Le premier triplet est celui du propriétaire du fichier. Il dispose d'un jeu de droits d'accès pour lui seul. En général, le propriétaire a tous les droits sur le fichier.
- Le deuxième cercle concentrique est celui du groupe. Cela permet de donner un certain nombre de droits d'accès à un groupe dont fait partie le propriétaire.
- Le cercle extérieur est celui de tous les autres utilisateurs. Les droits de ce troisième cercle sont ceux qui s'appliquent à tous les utilisateurs non privilégiés.

Au départ, ces trois groupes de trois droits peuvent te sembler complexes. Il faut savoir que les droits ont été inventés au départ pour contrôler l'utilisation des données sur de gros ordinateurs. Dans un grand système, il est indispensable de pouvoir rendre invisibles certains fichiers, d'en partager d'autres avec seulement certaines personnes, pour n'en laisser que certains en libre-service.

Sur un nano-ordinateur comme ton Raspberry, l'existence de ces droits d'accès peut te compliquer la vie. Souvent, il n'y a qu'un compte utilisateur, le tien. Autrement dit, tu dois *a priori* avoir tous les droits sur tous les fichiers, non ?

Pas vraiment. Il faut savoir que sous Linux, les applications sont aussi des utilisateurs. Grâce aux droits d'accès, tu vas pouvoir interdire à certaines applications de modifier des fichiers avec lesquels elles n'ont rien en commun.

De plus, bien gérer les droits d'accès va te devenir indispensable si tu comptes utiliser ton Raspberry comme serveur Web accessible sur Internet. Les droits vont te servir à renforcer ta sécurité face aux pirates.

Enfin, les droits d'accès te protègent de certaines erreurs de manipulation, comme la suppression par mégarde d'un fichier indispensable.

Afficher les droits d'accès

Pour un premier contact avec les droits d'accès, ouvre une fenêtre de terminal pour accéder à l'invite de commande du système. Saisis ensuite la commande suivante en validant par entrée :

```
ls -l
```

Tu dois voir apparaître une liste de noms de fichiers. Observe bien la première colonne qui contient des lettres (Figure 11.1). Le contenu exact de l'affichage sera sans doute différent sur ton ordinateur.

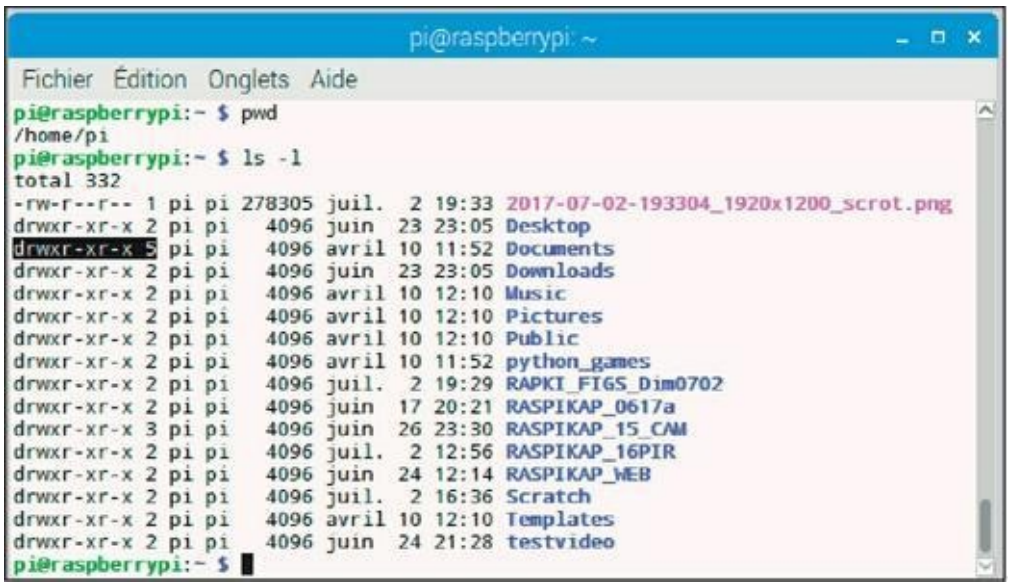


Figure 11.1 : Affichage des droits d'accès dans un répertoire.

La séquence de lettres et de tirets de la première colonne est la liste des droits d'accès. Elle contient 10 positions :

```
rwxrwxrwx
```

Lorsqu'il y a un tiret, le droit d'accès correspondant n'est pas défini. Lorsqu'il y a une lettre, il est défini en conséquence.

Pour la plupart des fichiers, il y a beaucoup de positions avec un tiret, comme ceci :

```
-rwxrw-r--
```

Décrypter les droits d'accès

Cette série de 10 lettres et tirets n'est pas très simple à lire, non ? En fait, c'est une sorte de code, qui n'est pas si difficile à décrypter.

Tout d'abord, la première position permet de savoir si la ligne concerne un répertoire ou un fichier. Lorsque c'est un répertoire, il y a à cet endroit-là un d comme *Directory*. C'est donc le nom d'un

sous-répertoire du répertoire courant. Tu peux y entrer avec cd ou afficher le contenu.

Pour être exact, la première position ne correspond pas à un droit d'accès. Tu ne peux pas changer cette position. Elle a été ajoutée avant les trois groupes de trois lettres de droits d'accès parce que c'était pratique à cet endroit.

Les trois lettres rwx qui suivent correspondent au premier triplet de droits d'accès à ce fichier ou répertoire. Prenons un autre exemple :

```
rw-
```

Ces trois positions signifient que la lecture est autorisée, que l'écriture est autorisée, mais que l'exécution ne l'est pas.

Il y a en tous trois groupes de trois positions pour les droits en lecture, en écriture et en exécution.

- Le premier groupe de trois concerne les droits d'accès du propriétaire du fichier du répertoire.
- Les trois positions suivantes correspondent aux droits d'accès pour le groupe.
- Les trois derniers éléments concernent les droits pour tous les autres utilisateurs de l'ordinateur.

Essayons maintenant de décrypter le groupe de droits d'accès suivant :

```
drwxrw-r--
```

Il faut d'abord subdiviser cette chaîne en quatre portions :

```
d   rwx   rw-   r--
```

Il ne reste plus qu'à décrypter les droits d'accès de chaque sous-ensemble :

Il s’agit ici d’un répertoire, puisqu’il y a la lettre d en première position.

Le propriétaire du fichier peut lire, écrire et donc modifier et même exécuter (rwx).

Le groupe peut lire et écrire, mais pas exécuter (rw-).

Les autres utilisateurs ne peuvent que lire le fichier (r--).

Les utilisateurs et les groupes

Comme déjà dit, dans Linux, chaque dossier et chaque répertoire appartient à un utilisateur. En général, il s’agit de celui qui a créé le fichier ou le répertoire.

Si tu as ouvert une session en tant qu’utilisateur portant le nom `pi`, tu es propriétaire de tous les fichiers que tu crées dans ton répertoire de travail. Quasiment tous les autres fichiers sont la propriété du super-utilisateur `root`. Les fichiers sont ainsi protégés contre les attaques des pirates et contre les erreurs de commande des utilisateurs normaux.

Faut-il en conclure que la plupart des fichiers de ton système ne t’appartiennent pas ? Exactement. En tant qu’utilisateur normal, avec le nom `pi`, il y a de nombreux fichiers que tu ne peux ni modifier, ni exécuter. Lorsque tu as besoin d’intervenir sur un des fichiers qui appartient au système, tu dois utiliser en préfixe la commande sudo pour basculer temporairement en mode super-utilisateur avec tous les droits.

Les groupes d’utilisateurs

Chaque fichier appartient également à un groupe qui est un ensemble d’utilisateurs. Les groupes sont très utiles dans les grands systèmes. Si tu travailles par exemple sur un projet avec d’autres personnes, le

fait d'appartenir au même groupe te permet de partager tes fichiers uniquement avec les autres membres de ton groupe.

Sur un nano-ordinateur tel que le Raspberry, il y a au moins un groupe, mais il n'a qu'un seul utilisateur au départ, c'est le groupe `pi` qui possède l'utilisateur `pi`. Il y a également un groupe `root` pour le super-utilisateur `root`. Autrement dit, dans ta situation, tu peux ignorer les droits d'accès des groupes.

Mais il faut savoir qu'une application est aussi un utilisateur. Elle n'ouvre pas une session en saisissant un nom et un mot de passe au clavier, mais elle possède un nom et appartient à un groupe ou à plusieurs. Une application peut ainsi être propriétaire de certains répertoires.

Les groupes permettent d'abord de partager des fichiers entre plusieurs applications qui ont besoin de fonctionner de concert. Toutes les applications qui ont à utiliser la messagerie peuvent ainsi partager des fichiers dans le groupe `mail`. De même, les applications qui participent au fonctionnement de ton ordinateur appartiennent au groupe `staff`, dans lequel se trouve également le super-utilisateur.

En temps normal, tu n'auras pas besoin de t'occuper de ces détails. Dans de rares cas, il te faudra modifier les droits d'accès du groupe pour permettre à une application de fonctionner ou d'interagir avec une autre.

Savoir qui est l'utilisateur et le groupe

Dans la Figure 11.2, tu peux apercevoir une liste de deux fichiers tels qu'ils sont trouvés dans un des sous-répertoires utilisés normalement par le super-utilisateur. Après la série des lettres des droits d'accès, tu peux lire deux noms. Le premier est le nom du propriétaire et l'autre celui du groupe.

Il est donc facile de savoir à qui appartient chaque fichier et répertoire.

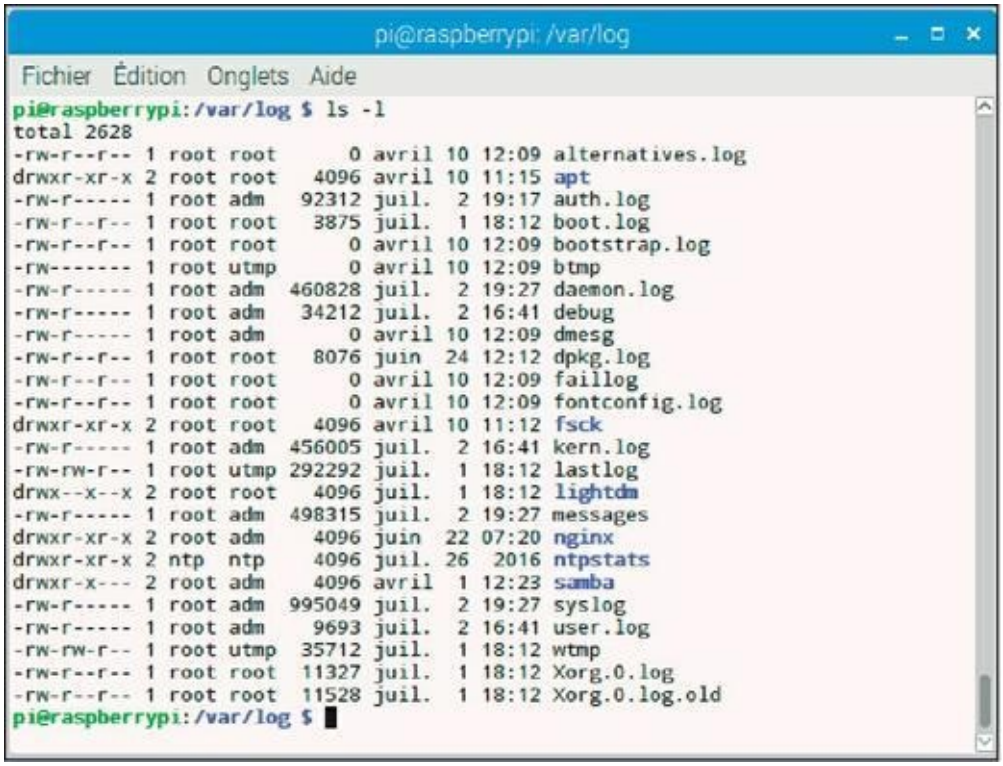


Figure 11.2 : Affichage du nom du propriétaire et du groupe.

Les autres utilisateurs

Le dernier groupe de trois lettres de droits d'accès concerne les autres utilisateurs, c'est-à-dire les personnes non privilégiées. Supposons que tu viens de rassembler de jolies photographies de chatons et que tu désires les partager avec tout le monde. Il suffit d'activer le droit d'accès en lecture dans le triplet des droits pour tous les utilisateurs. Tes jolies images seront accessibles immédiatement dans tout le réseau.

Si tu décides de ne plus rendre ces images disponibles à tout le monde, il suffit de désactiver le droit en lecture pour tous les utilisateurs.

Lorsque l'utilisateur `pi` n'est pas privilégié pour un fichier, c'est-à-dire qu'il fait partie du dernier triplet, ses droits d'accès sont limités.

Tu ne peux évidemment pas modifier les fichiers qui appartiennent au super-utilisateur. Les fichiers qui lui appartiennent n'offrent aucun

accès au groupe des autres utilisateurs, notamment en écriture. Cela t'évite d'en effacer un par mégarde.

Lorsque tu as besoin de réparer quelque chose, il te faut donc utiliser le préfixe de commande sudo pour basculer en mode super-utilisateur. Dans ce mode, c'est comme si les fichiers du système étaient devenus les tiens.

Pour être exact, ce sont les droits d'accès du propriétaire qui te sont accordés dans ce mode.

Pour modifier les droits d'accès

Tu disposes d'un certain nombre de commandes pour gérer les droits d'accès (Tableau 11.1).

Tableau 11.1 : Quelques commandes pour les droits d'accès.

Commande	*Description*
ls -l	Affiche les fichiers avec les droits d'accès.
chmod	Modifie les droits d'accès.
chown	Modifie le propriétaire.
groups	Affiche les utilisateurs d'un groupe.
useradd	Ajoute un utilisateur à un groupe.
chgrp	Modifie le groupe d'un fichier.

Tu sais déjà utiliser l'option -l de la commande de listage ls pour afficher les droits d'accès des fichiers d'un répertoire. Mais comment faire pour intervenir sur ces droits d'accès ?

La commande chmod

La commande chmod permet de modifier les droits d'accès. Son nom doit être suivi de trois paramètres :

- une indication de la cible (le propriétaire, le groupe ou les autres) ;
- les modifications de droit à apporter, selon différentes techniques ;
- le type de droit à modifier.

Voici la présentation générique de la commande :

```
sudo chmod a+w NomFichier ou NomRépertoire
```

Nous allons détailler cette commande dans la suite. Tiens bien compte du fait qu'il faut toujours utiliser le préfixe sudo pour basculer en mode super-utilisateur afin de pouvoir modifier les droits.

Indication de la cible

Le tableau suivant présente les lettres que tu peux mentionner pour désigner le bénéficiaire du changement de droit.

Tableau 11.2 : Cibles des changements de droits.

Lettre	*Description*
u	Le propriétaire du fichier.
g	Le groupe.
o	Tout le monde sauf le propriétaire et le groupe.

a	Absolument tout le monde.

Choix du mode de modification

Le deuxième paramètre sert à choisir quel genre de modification tu veux apporter.

Tableau 11.3 : Modes de changement de droits.

Lettre	*Description*
+	Ajoute ou active un droit.
-	Enlève ou désactive un droit.
=	Ignore les droits actuels et en définit de nouveaux.

Les deux premières options s'appliquent à des droits qui existent déjà et permettent d'en ajouter ou d'en enlever. Cela permet par exemple de modifier le droit en écriture sans changer le droit en lecture et en exécution.

La troisième option, =, sert à modifier tous les droits d'un coup. Elle est utile lorsque tu n'as pas besoin de tenir compte de l'état actuel des droits.

Choix du droit à modifier

Le Tableau 11.4 présente les quatre lettres que tu peux indiquer pour désigner un droit. Ce sont les lettres affichées lorsque tu listes les fichiers d'un répertoire.

Tableau 11.4 : Désignation des droits d'accès.

Lettre	*Description*

r	Droit de lecture, read.
w	Droit d'écrire, write.
x	Droit d'exécuter, execute.
X	Droit d'exécution spécial pour les répertoires.

La signification de ces droits ne t'est plus totalement étrangère, mais nous pouvons faire quelques remarques concernant les droits en exécution :

- Tu ne peux afficher le contenu d'un répertoire que si tu possèdes les droits en exécution. Il ne suffit pas d'avoir un droit en lecture, bizarrement.
- Tu dois avoir les droits en exécution sur un fichier pour pouvoir changer son nom. Les droits en écriture ne suffisent pas.
- Si tu as le droit en lecture pour un fichier, tu peux le faire exécuter s'il est destiné à être lu par un programme. Dans ce cas, tu n'as pas besoin de droit en exécution, puisque ce n'est pas toi qui vas l'exécuter, mais le programmes.

Tu peux par exemple lancer l'exécution d'un fichier de code source Python même si tu n'as que des droits en lecture sur ce fichier. En effet, c'est Python qui va lire le fichier puis l'exécuter. Ce sont les droits de Python sur le fichier qui comptent.

En revanche, si le fichier est le code binaire d'une application, tu dois avoir les droits en exécution pour le lancer.

Tout cela est assez complexe, je dois le reconnaître. Les cas spéciaux méritent de prendre le temps de bien les comprendre. Si tu oublies un

cas particulier, reviens à cette section du livre pour te rafraîchir la mémoire.

Lorsque tu vas avoir besoin d'utiliser plusieurs logiciels en combinaison, par exemple Python pour créer une page Web, il peut arriver que les choses ne fonctionnent pas. Dans ce cas, commence toujours par vérifier les droits d'accès.

Il arrive en effet que les droits d'accès empêchent le fonctionnement, mais sans afficher aucun message d'erreur. Cela ne marche pas, un point c'est tout. Prends donc l'habitude de vérifier les droits dès que quelque chose cloche et que tu n'as pas d'autres indices.

Récapitulons en pratique

Les droits d'accès sont un sujet complexe. Rien ne vaut une mise en pratique pour les mémoriser.

Partons d'un exemple simple. Supposons que nous voulions régler les droits d'accès en sorte que tous les comptes utilisateurs de notre nano-ordinateur aient le droit de créer un fichier. Est-ce que tu peux imaginer comment écrire la commande correspondante ? Nous supposons que les droits en lecture ont déjà été donnés à tous sur le fichier concerné.

Voici comment pourrait s'écrire la commande :

```
sudo  chmod  a+w  nomfichier_ou_chemin
```

La mention `chemin` dans l'exemple désigne un chemin d'accès à un sous-répertoire, comme `/home/pi`. Ce chemin est absolu s'il commence par la barre oblique, et relatif sinon. Dans ce cas, il part du répertoire courant.

La Figure 11.3 montre la situation avant et après changement des droits d'accès. À la fin de cette séquence, tous les comptes utilisateurs peuvent modifier le fichier concerné.

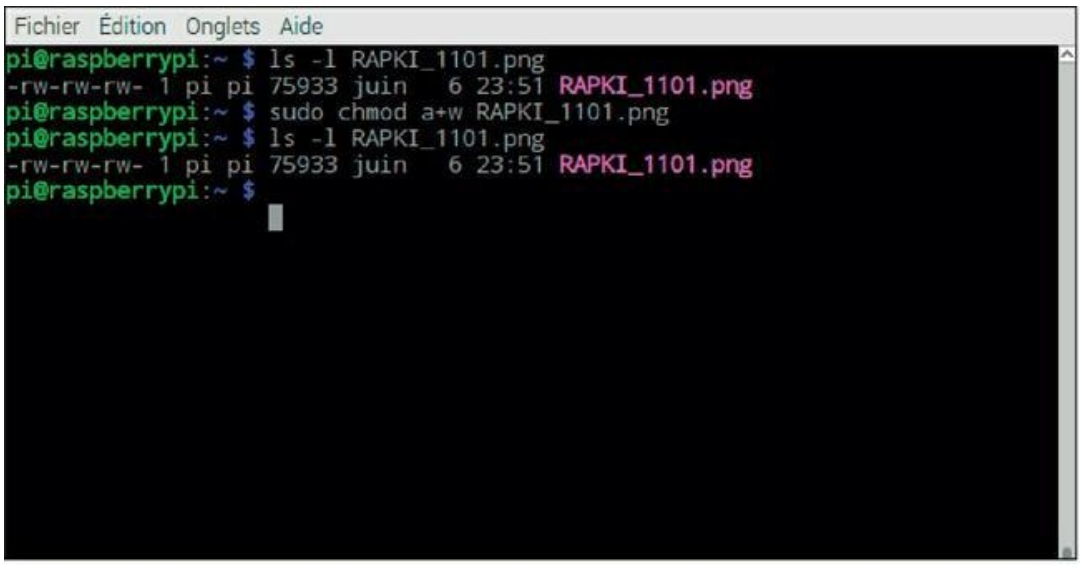


Figure 11.3 : Changement des droits d'accès à un fichier.

Pour modifier plusieurs droits en une seule fois, il suffit de les réunir comme ceci :

```
sudo chmod a+rwx nomfichier_ou_chemin
```

Il n'est pas obligatoire d'utiliser sudo en préfixe lorsque tu veux modifier les droits d'accès à des fichiers qui t'appartiennent. Tu peux faire ce que tu veux avec les fichiers qui se trouvent dans ton répertoire personnel. En revanche, c'est obligatoire dès que tu interviens ailleurs dans le système de fichiers de ta machine.

Droits d'accès par code numérique

Tu peux aussi utiliser des valeurs numériques pour régler les droits d'accès. Les magazines et les sites Web qui s'intéressent au sujet donnent parfois des conseils étranges du style : « Il faut modifier les droits d'accès au fichier avec la valeur 777 ou 644 » .

Ce codage numérique est une technique plus rapide pour régler les droits d'accès. Un tel code est plus facile à mémoriser qu'une longue

séquence de neuf lettres et tirets, et plus rapide à saisir.

Il reste à savoir à quoi correspondent ces valeurs. Facile : le premier chiffre concerne les droits du propriétaire, le deuxième les droits du groupe et le troisième les droits des autres. Ces trois chiffres correspondent donc à neuf lettres.

Le Tableau 11.5 montre comment trouver la valeur numérique d'un triplet de lettres.

Tableau 11.5 : Codes numériques des droits d'accès.

Nombre	*Lecture R*	*Écriture W*	*Exécution X*
7	r	w	x
6	r	w	-
5	r	-	x
4	r	-	-
3	-	w	x
2	-	w	-
1	-	-	x
0	-	-	-

Quelques exemples :

```
744 = rwxr--r--
777 = rwxrwxrwx
600 = rw-------
```

Tu peux utiliser ce code numérique à la place des lettres dans la commande chmod :

```
sudo chmod 644 nomfichier
```

La précédente commande équivaut à la nouvelle séquence de droits suivante :

```
rw-r--r--
```

L’option collective -R

Lorsque tu as besoin de modifier les droits de tous les fichiers d’un répertoire, il faut travailler fichier par fichier.

Pour éviter ce travail exténuant, il suffit d’ajouter l’option -R qui applique les nouveaux droits à tous les fichiers trouvés dans un répertoire. Il suffit d’ajouter cette option n’importe où dans la ligne de commande, avec une espace avant et une autre après.

Gestion des utilisateurs et des groupes

Pour bien exploiter les droits d’accès, il faut savoir utiliser les deux commandes chown et chgrp pour prendre le contrôle des propriétaires et des groupes.

La commande chown permet de modifier le propriétaire ou le groupe d’un fichier. Comme c’est une commande système, il faut basculer en mode super-utilisateur avec le préfixe sudo :

```
sudo chown nouveau_prop:nouveau_groupe
nomfichier_ou_chemin
```

Tu peux indiquer soit le nom du nouveau propriétaire avant le signe deux-points, soit le nom du nouveau groupe après le signe, ou les deux.

Voici par exemple comment transférer un fichier qui t'appartient au super-utilisateur :

```
sudo chown root:root nomfichier_ou_chemin
```

Note bien qu'il n'y a qu'un seul g dans le nom de la commande chgrp.

La commande groups

Cette commande permet de voir quels sont les groupes auxquels appartient un utilisateur. Si tu saisis le nom de la commande seul, tu vois apparaître les groupes auxquels tu appartiens (*a priori* l'utilisateur `pi`).

La Figure 11.4 montre les groupes auxquels appartiennent les deux utilisateurs `pi` et `root`. Tu vois que tu appartiens dès le départ à un certain nombre de groupes.

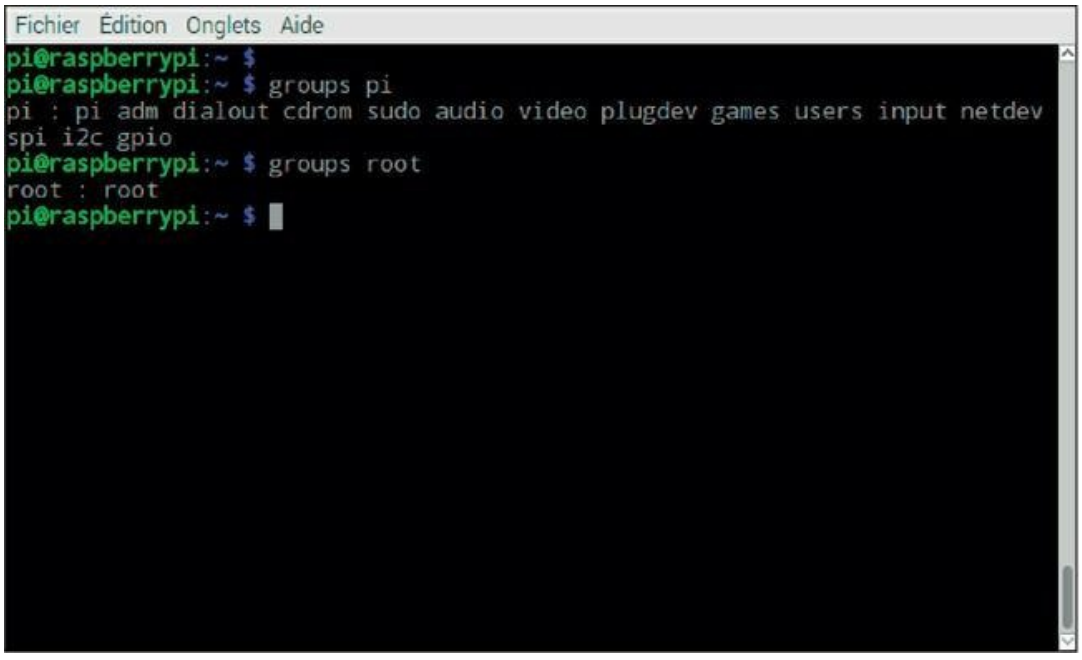


Figure 11.4 : Utilisation de la commande groups.

Ce grand nombre de groupes permet dès le départ de pouvoir utiliser la plupart des logiciels installés dans le système, sans avoir besoin de venir rejoindre le groupe correspondant. C'est notamment important pour les deux groupes portant le nom `audio` et `video`. Si l'utilisateur `pi` n'appartenait pas à ces deux groupes, il te serait impossible d'utiliser une caméra ou de faire jouer de la musique sur la sortie casque.

Ajouter un utilisateur à un groupe

La commande useradd sert à ajouter un utilisateur à un groupe.

```
sudo useradd -G nomgroupe nomutil
```

N'oublie pas l'option -G qui doit toujours être suivie du nom du groupe. En général, tu utiliseras en préfixe la fameuse commande sudo.

Tu auras rarement besoin de le faire, mais tu peux créer un nouveau groupe avec la commande groupadd :

```
sudo groupadd nomgroupe
```

Ici aussi, il faut utiliser le préfixe sudo.

Bien gérer ses fichiers

Une fois que tu sais comment accéder à des fichiers, tu peux t'intéresser à leur gestion en mode texte. En effet, tant que tu ne connais pas le mécanisme des droits d'accès, tu risques de faire souvent face à des erreurs du type « Permission denied » , ce qui signifie que tu tentes d'accéder à un fichier ou à un répertoire pour

lequel tu n’as pas suffisamment de droits d’accès. C’est notamment le cas lorsque tu sors de ton répertoire personnel `home`.

L’application Gestionnaire de fichiers disponible sur le bureau en mode graphique permet d’intervenir sur les droits d’accès aux fichiers en ouvrant le menu local par clic-droit dans le nom de fichier puis en choisissant `Propriétés`. Tu peux alors ouvrir la seconde page, `Per missions` ou `Droits d’accès` et intervenir sur les trois listes déroulantes. C’est cependant beaucoup moins rapide qu’avec une commande en mode texte dans le terminal, surtout lorsqu’il faut travailler sur plusieurs fichiers à la fois.

Pour créer un fichier vide (touch)

Tu peux créer un fichier vide au moyen de la commande touch.

```
touch  nouveaunom
```

Voici un exemple :

```
touch  monmemo.txt
```

Si tu utilises la commande dans ton répertoire personnel, le fichier qui va être créé t’appartiendra. La chaîne de droits d’accès ressemblera à la suivante :

```
-rw-r--r--
```

Tu peux donc lire et écrire ce fichier. En revanche, il n’est pas exécutable, ce qui est normal si ce n’est pas une application. Les autres utilisateurs ne peuvent que le lire.

À chaque fichier correspond une heure de dernière modification. Tu peux changer cette heure simplement en utilisant la commande touch sur le fichier existant, ce qui est d’ailleurs à l’origine de son nom.

Pour créer un fichier du super-utilisateur

S'il t'arrive un jour de devoir intervenir dans les entrailles du système Linux, il te faudra savoir comment créer un fichier dont le propriétaire est le super-utilisateur. Voici comment faire :

```
sudo  touch  nouveaufic_systeme
```

Le fichier qui résulte de la commande va appartenir à `root` et au groupe de même nom. Les droits d'accès ne changent pas, mais ce sont dorénavant les droits du super-utilisateur, qui est le seul à pouvoir modifier le fichier. :

```
-rw-r--r--
```

Si tu essaies de modifier le fichier en tant qu'utilisateur normal, tu obtiendras un message de refus. Ce sont les trois dernières lettres du masque de droits (r--) qui s'appliquent dans ce cas.

Si tu prends suffisamment de précautions, tu peux éviter d'être ainsi limité dans tes possibilités de modification en t'octroyant des droits d'accès de la façon suivante :

```
sudo  chmod  a+w  nouveaufic_systeme
```

Dorénavant, tu peux modifier ce fichier. Le principal avantage de cette astuce est de permettre une modification du contenu de façon plus confortable dans l'interface graphique, en chargeant le fichier dans l'éditeur Leaf (il se lance dans les `Accessoires`, sous le nom `Text editor`).

Pour copier un fichier ou un répertoire (cp)

La commande permettant la copie porte le nom `cp` :

```
cp  ancien_nom_ou_chemin
nouveau_nom_ou_chemin
```

Tu n'as pas besoin de t'occuper du chemin d'accès si tu veux faire une copie dans le répertoire courant.

Lorsque tu fais une copie d'un fichier, le contenu du fichier est évidemment le même que l'original. En revanche, tu deviens propriétaire de la copie, avec le masque de droits suivant :

```
-rw-r--r--
```

Autrement dit, à partir du moment où tu peux faire une copie d'un fichier qui appartient au super-utilisateur, la copie devient la tienne. C'est utile dans certains cas.

Pour pouvoir copier un fichier vers un autre répertoire, tu dois avoir les droits d'accès en écriture sur ce répertoire.

Pour renommer un fichier ou un répertoire (mv)

La commande de déplacement mv (move) s'utilise de la façon suivante :

```
mv  ancien_nom  nouveau_nom
```

Le nom est l'abréviation du terme anglais *move* qui signifie déplacer. La commande permet donc également de déplacer un fichier depuis le répertoire courant vers un autre :

```
mv  nom_fichier  chemin
```

Pour supprimer un fichier ou un répertoire (rm)

Pour supprimer un fichier, tu disposes de la commande rm qui est l'abréviation de *remove*.

```
rm  nom_fic
```

Bien sûr, tu dois avoir les droits en écriture pour pouvoir supprimer le fichier. Si nécessaire, tu peux forcer le passage avec le préfixe sudo.

Pour supprimer un sous-répertoire, tu disposes de l'option -r (ou -R) :

```
rm  –r  nom_repertoire
```

Il faut savoir que cette commande supprime la totalité du contenu du répertoire, et donc tous ses sous-répertoires. Tu dois donc être vigilant.

Pour te protéger, Linux refuse de supprimer un répertoire s'il n'est pas vide. Tu peux passer outre cette protection en ajoutant l'option -f qui permet de forcer :

```
sudo  rm  –rf  nom_repertoire
```

Cette commande est une arme de destruction massive. Elle supprime tout ce qui est trouvé dans le répertoire et dans tous les sous-répertoires, et ce de façon irrémédiable. Il n'y a même pas de demande de confirmation. Tu ne te serviras donc de cette commande que si tu es vraiment sûr de toi.

Les méta-caractères

Souvent, on a besoin d'appliquer la même commande à tous les fichiers se trouvant dans un répertoire ou à tous ceux du même type. Linux propose à cet effet des caractères génériques ou méta-caractères. Le plus important est l'astérisque.

Pour changer la date et l'heure de tous les fichiers du répertoire courant, il suffit d'indiquer l'astérisque à la place du nom de fichier :

```
touch  *
```

Tu peux limiter la commande à tous les fichiers portant la même extension, c'est-à-dire qui sont du même type. Voici comment changer la date de tous les fichiers de code source Python, qui portent l'extension `.py`:

```
touch  *.py
```

Tu peux bien sûr intervenir sur les fichiers d'un autre répertoire, en ajoutant le chemin d'accès :

```
touch  /home/autrecompte
```

Ce méta-caractère est utilisable avec les autres commandes comme celle de copie et celle de déplacement.

Pour installer un logiciel

Tu sais maintenant que la gestion des fichiers en mode texte n'est pas simple au départ sous Linux. Tu peux donc te demander si l'installation d'une application n'est pas encore plus difficile.

En fait, c'est plus simple que sous Windows.

En effet, sous Linux, tu disposes d'un outil appelé gestionnaire de paquetages. Il se charge d'installer correctement tous les logiciels et fichiers nécessaires au logiciel principal que tu as demandé d'installer. Tous les fichiers nécessaires sont regroupés dans une archive que l'on appelle un paquetage.

Lorsqu'un logiciel a besoin d'un autre pour fonctionner, cet autre logiciel est une *dépendance*.

Dans les deux variantes de Linux que sont Raspbian et Debian, le gestionnaire de paquetages porte le nom `apt-get` et s'utilise de la façon suivante :

```
sudo apt-get install nompaquetage
```

Tu trouveras le nom exact du paquetage dans un magazine ou sur le Web, ou en te renseignant auprès d'un ami.

Pendant l'installation, l'outil te demandera souvent de répondre par Y, O ou N pour poursuivre. Pour éviter ces questions, il suffit d'ajouter l'option -y.

```
sudo apt-get install -y nompaquetage
```

Pendant l'installation, tu vas voir apparaître un grand nombre de messages que tu peux ignorer.

En revanche, ce qui est absolument indispensable, c'est un accès correct à Internet. Tous les logiciels sont en effet téléchargés depuis des sites Web appelés *référentiels* ou *repositories*.

Enfin, il faut absolument utiliser la commande préfixe sudo pour installer un logiciel.

Pour mettre à jour et à niveau

Le système conserve une liste de toutes les dépendances entre les logiciels sur la machine locale. Mais pendant ce temps, les logiciels évoluent sans cesse. Des corrections sont apportées et de nouvelles fonctions ajoutées.

Pour remettre à jour toutes les dépendances en un seul geste, tu choisis la sous-commande update :

```
sudo apt-get update
```

Cette commande est à lancer avant chaque installation d'un logiciel. Si tu l'oublies, ce n'est pas très grave, mais il est toujours conseillé de partir d'une liste la plus à jour possible.

Pour lancer une mise à jour de tous les logiciels actuellement installés, utilise la sous-commande upgrade :

```
sudo apt-get upgrade
```

Comme tu t'en doutes, l'exécution de la commande peut prendre un certain temps. Mais tu n'as pas besoin de la lancer souvent. La Figure 11.5 montre un exemple de mise à jour. Il faut parfois frapper la touche Y puis Entrée pour confirmer.

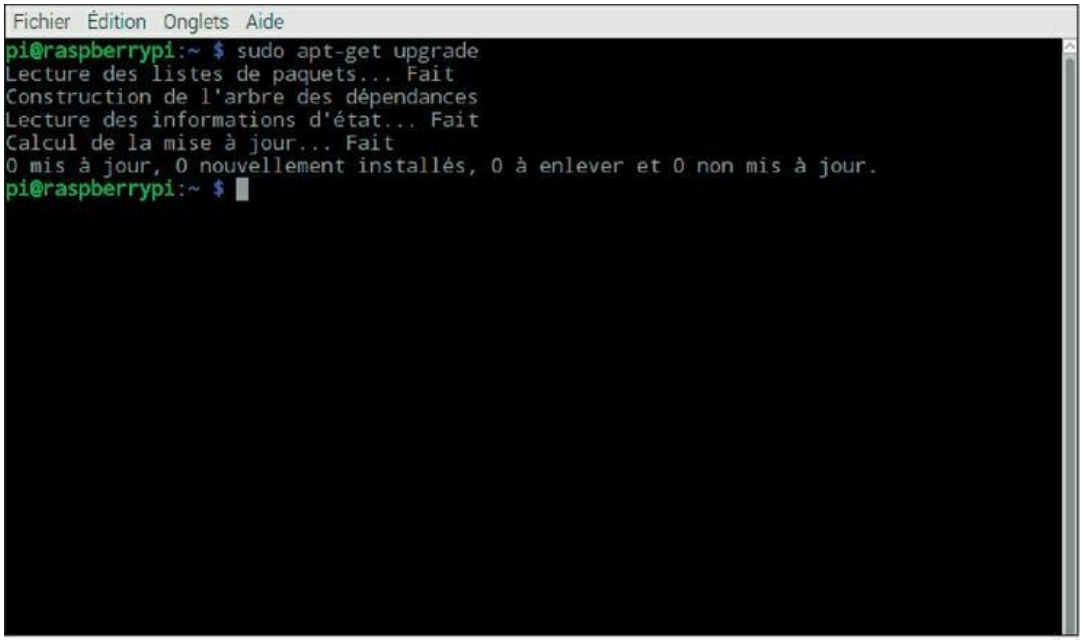


Figure 11.5 : Session de mise à jour d'applications.

Désinstaller un logiciel

Tu auras rarement besoin de désinstaller un logiciel sur ton Raspberry Pi, car l'espace sur ta carte mémoire devrait largement suffire.

Voici cependant les deux commandes utilisables pour supprimer tout un paquetage :

```
sudo apt-get remove nom_paquetage
sudo apt-get purge  nom_paquetage
```

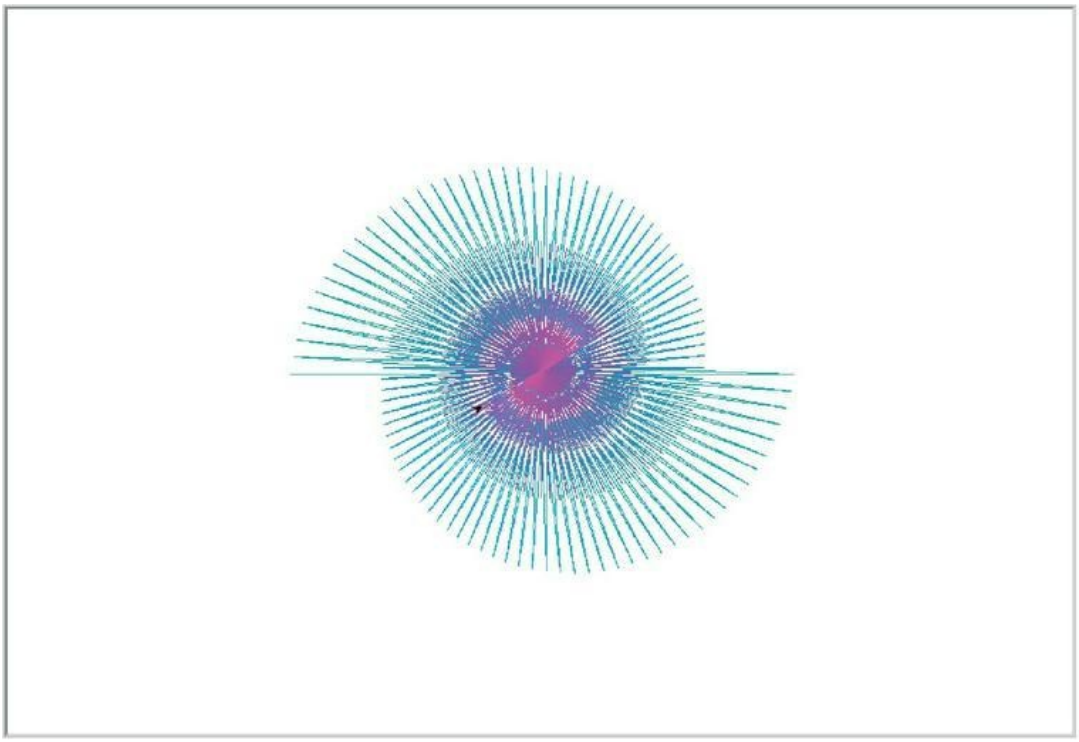
La première commande supprime le logiciel lui-même, alors que la seconde supprime aussi tous ses fichiers de configuration. Elle sera utile lorsque tu auras l’impression qu’un problème a rendu ces fichiers inutilisables. Dans ce cas-là, mieux vaut repartir de zéro.

Des projets distrayants

Au programme de cette partie :

Chapitre 12
La tortue dessinatrice

Le langage Python est livré avec une tortue. Non, ce n'est pas une vraie tortue, et tu n'auras pas besoin de la nourrir. En revanche, cette tortue est douée pour le dessin.



Bonjour, tortue Python !

La tortue Python est en fait un robot dessinateur que tu diriges dans l'écran pour qu'elle trace des figures. Pour la faire dessiner, tu lui

envoies des commandes en langage Python. Avec un peu de pratique, tu vas pouvoir produire des images d'une grande beauté.

En termes techniques, la tortue est une famille de fonctions réunies dans une librairie. Tu dois citer cette librairie dans ton programme Python pour pouvoir utiliser ses fonctions. Voici la ligne qu'il faut ajouter au début de chaque projet pour inviter la tortue :

```
import turtle
```

À partir de ce moment, tu peux utiliser les commandes pour contrôler la tortue. Quand tu lances l'exécution du projet, tu vois apparaître une troisième fenêtre graphique dans laquelle sont réalisés les tracés.

La Figure 12.1 donne un exemple de dessin assez complexe réalisé avec quelques commandes.

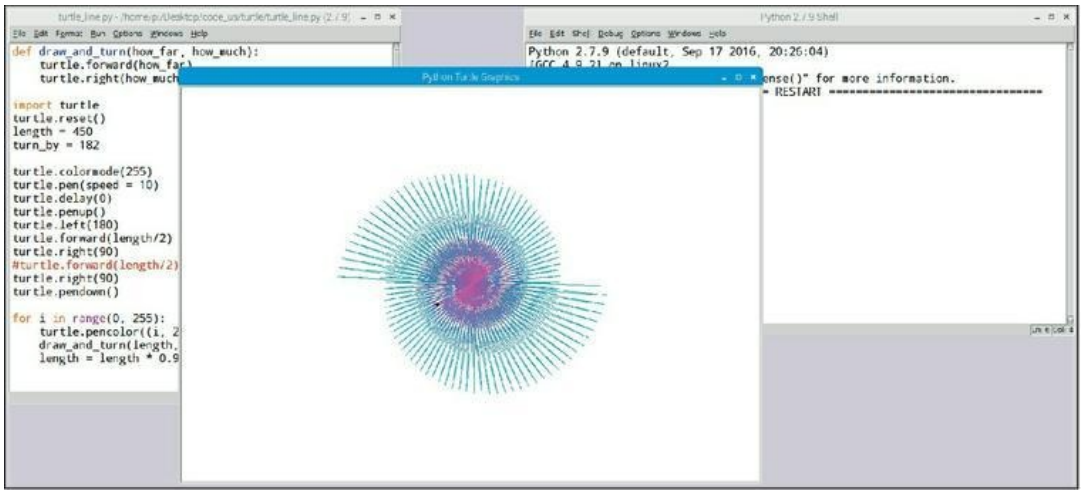


Figure 12.1 : Exécution d'un programme d'exemple.

Découvrir les commandes

Tu vas constater que contrôler la tortue est vraiment simple, tout d'abord parce qu'il n'y a que trois sortes de commandes :

La première catégorie sert à diriger et orienter la tortue.

La deuxième catégorie sert à la faire avancer en laissant une trace (ou pas).

La troisième catégorie sert à contrôler la levée de la plume, le changement de couleur de trait et autres opérations complémentaires comme l'effacement de l'écran.

Dans la plupart des projets de dessin avec la tortue, vient d'abord un groupe de commandes pour préparer le tracé en définissant quelques valeurs et formes simples. Vient ensuite une autre série de commandes pour effectuer le tracé réel.

Tout cela semble assez simple, non ?

Bien se préparer

En plus de faire connaître à Python ton intention d'utiliser la librairie de fonctions de la tortue, `turtle`, il est conseillé de la replacer au centre, afin de démarrer dans des conditions connues. Il suffit d'ajouter un appel à la fonction `reset()` comme ceci :

```
turtle.reset()
```

Tracer une ligne

La Figure 12.2 montre un premier tracé de ligne qui part du centre et va vers la droite. C'est le résultat de l'instruction suivante :

```
turtle.forward(200)
```

J'ai choisi de faire avancer la tortue de 200 unités, ce qui donne un résultat bien visible. Tu peux faire des essais avec d'autres valeurs pour voir la différence.

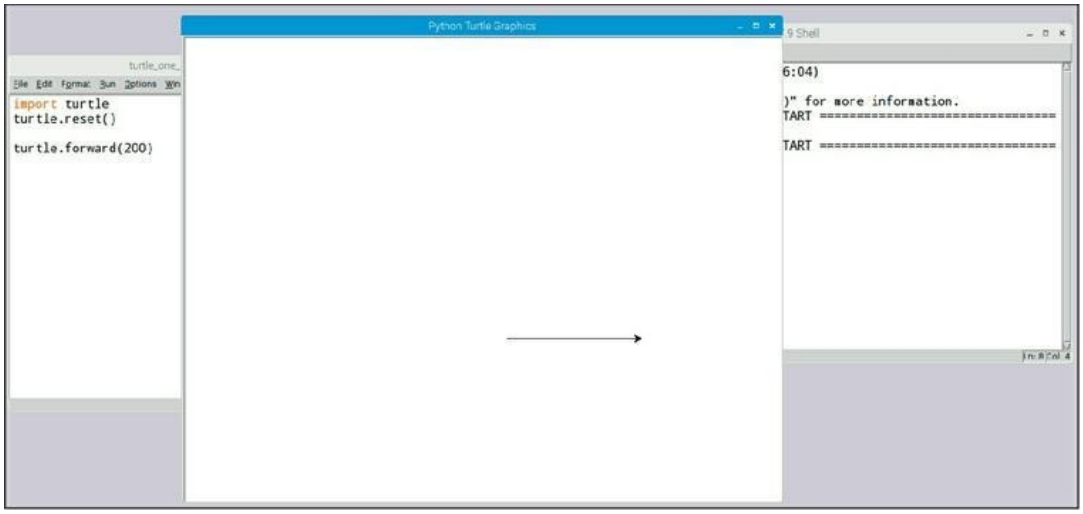


Figure 12.2 : Premier tracé de tortue.

Si tu demandes un tracé trop long, la tortue va tomber en dehors du bord de l'écran, ou plutôt de la fenêtre. Tu peux continuer à la contrôler, mais tu ne la vois plus. Dans ce cas, mieux vaut la replacer au centre avec `reset()`.

La tortue ne se limite pas à marcher dans un sens. Tu peux la faire reculer en indiquant une valeur négative ou bien utiliser la commande `backward()` :

```
# Ces deux lignes ont le même effet :
turtle.forward(-200)
turtle.backward(200)
```

Il est parfois plus pratique de n'utiliser que la marche en avant, car avec des calculs, tu peux obtenir des valeurs positives et négatives.

La tortue tourne !

Faire tourner la tortue est très simple. Il faut seulement savoir de quel angle tu veux la faire tourner. Les rotations sont exprimées en degrés. Pour faire un tour complet, tu utilises la valeur 360.

Le schéma de la Figure 12.3 montre les principaux angles de rotation possibles. Lorsque tu fais tourner la tortue, il faut imaginer que tu es assis dessus. Les angles sont toujours relatifs à l'orientation actuelle de la tortue.

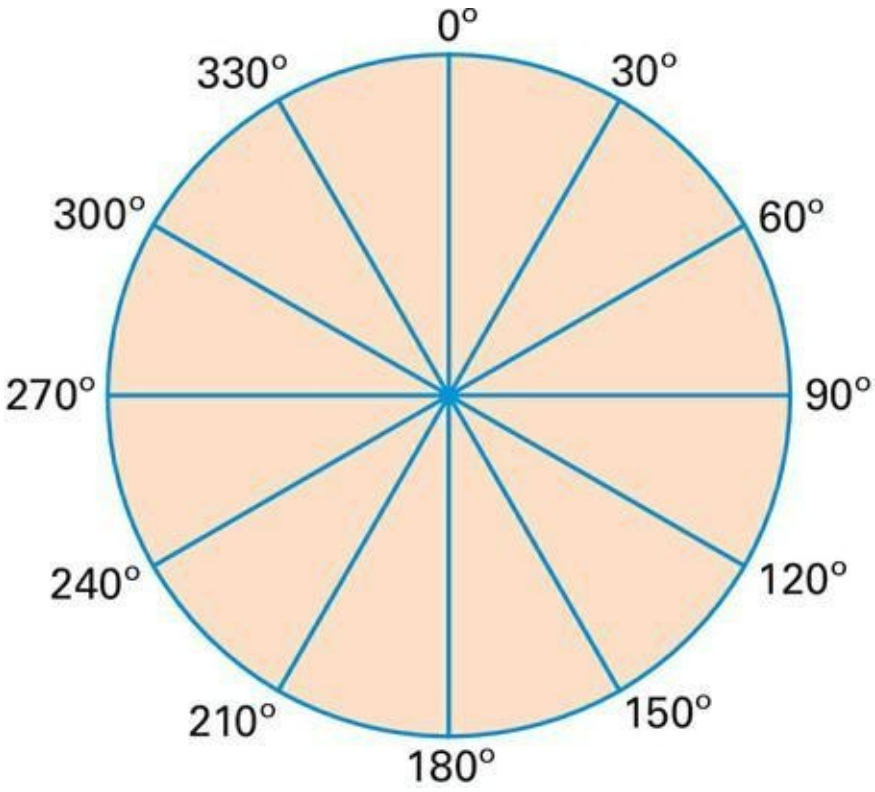


Figure 12.3 : La rose des vents de la tortue.

Tableau 12.1 : Quelques angles de rotation.

Angle	*Nouvelle direction*
0	Tout droit.
90	Quart de tour à droite.
180	Demi-tour complet.
270	Quart de tour à gauche.

360	Identique à zéro.

Voici comment faire un quart de tour à droite puis un autre à gauche :

```
turtle.right(90)   # Quart droite
turtle.left(90)    # Quart gauche
```

Le signe # est le début de commentaire. Tout ce qui le suit jusqu'à la fin de la ligne est ignoré par Python.

Tu n'es pas limité aux multiples de 90 ! Tu peux choisir n'importe quelle valeur entre 0 et 360, même si les multiples de 90 sont plus faciles à suivre. Lorsque tu as besoin de tourner de 275°, il n'est pas aussi simple de savoir dans quelle direction tu te retrouves, mais il n'y pas le choix: certains dessins réclameront de nombreuses valeurs intermédiaires.

Où sont la gauche et la droite ?

Les directions que tu donnes à la tortue sont relatives, comme dans les systèmes de navigation des automobiles. La voix te dit de prendre la prochaine droite, puis la deuxième à gauche puis tout droit.

Cela ne serait pas pratique si les indications étaient du style « d'abord vers l'ouest, puis vers le nord puis plein sud » .

Dans le langage de la tortue, le fait d'écrire `right(90)` signifie tourner à droite. Cela ne va pas obligatoirement vers l'est.

Pour tourner à gauche, tu peux soit donner une commande pour tourner à gauche de 90 ou à droite de 270. Essaie avec la tortue pour le vérifier.

Il est parfois plus simple de ne faire que des virages à droite. À toi de choisir comment tu veux piloter ton animal.

Un second trait

Tu peux faire tourner ta tortue sur elle-même autant de fois que tu le veux. Elle n'en aura jamais assez. En revanche, tourner ne veut pas dire tracer.

Pour tracer une autre ligne, il faut utiliser les commandes de déplacement en avant et en arrière.

Si tu ne tournes pas la tortue avant de faire un nouveau tracé, le nouveau trait va continuer au bout du précédent.

Si tu tournes la tortue avant d'effectuer le deuxième tracé, tu obtiendras un angle. La Figure 12.4 montre le résultat d'un premier trait suivi d'une rotation de 90 et d'un autre trait. Voici d'abord le code source complet :

Listing 12.1 : Code source de turtle_two_lines.py

```
import turtle
turtle.reset()

turtle.forward(200)
turtle.right(90)
turtle.forward(200)
```

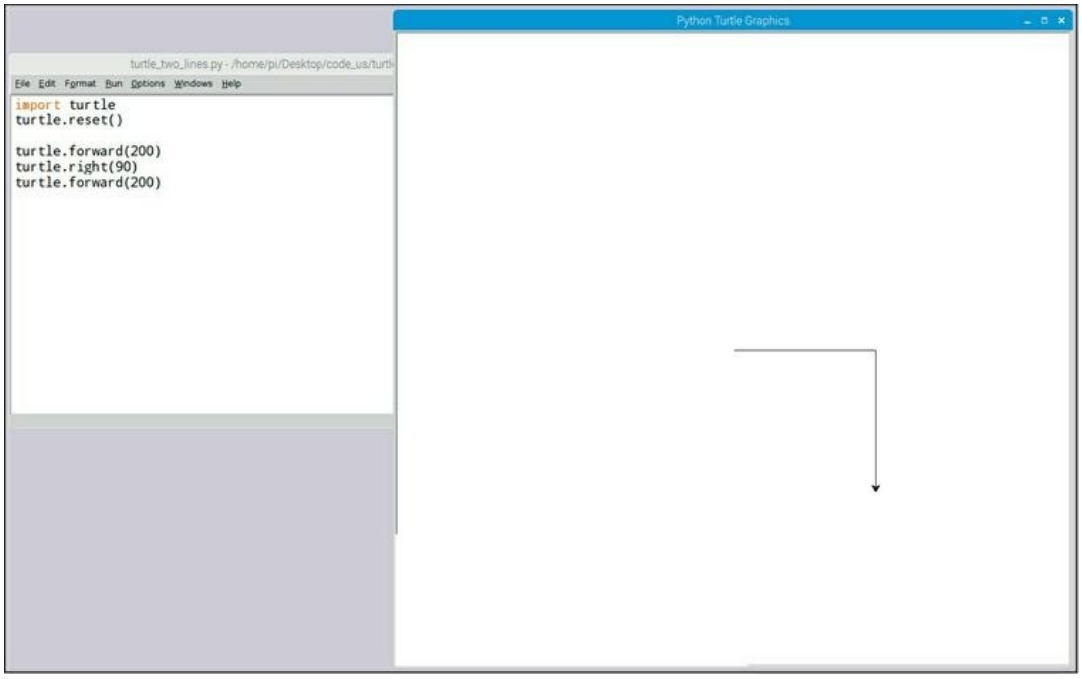


Figure 12.4 : Résultat de l'exemple turtle_two_lines.

Profite des versions abrégées des principales commandes : fd pour forward, bk pour backward, rt pour right et lt pour left.

Un premier dessin complet

Pour créer une figure, il suffit de réunir plusieurs tracés. Tu peux facilement produire des étoiles, des carrés, des triangles et autres formes géométriques simples uniquement à partir de déplacements et de rotations.

Dessiner un carré

Pour obtenir un carré, il suffit de se déplacer et d'effectuer quatre rotations. Voici le code source :

```
turtle.forward(200)
turtle.right(90)
```

```
turtle.forward(200)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
```

En fin de programme, la tortue est revenue à son point de départ, et orientée dans la même direction.

Ce programme fonctionne, mais il contient beaucoup de répétitions. De plus, si tu veux maintenant dessiner un carré plus petit, il faut changer quatre fois la longueur du tracé dans les commandes de mouvement.

Pourquoi ne pas mettre en place une boucle de répétition basée sur le mot-clé for ?. Voici comment l'utiliser :

Listing 12.2 : Code source de for_next_square.py

```
import turtle
turtle.reset()

for turns in range (0, 4):
    turtle.forward(200)
    turtle.right(90)
    print turtle.position()

turtle.setheading(0)
```

La Figure 12.5 montre le résultat de ce programme optimisé. À partir de ce moment, il suffit de changer une fois la longueur de tracé pour changer la taille du carré. La dernière commande repositionne la tortue.

Pour tracer un autre carré en miroir, tu peux remplacer la commande right(90) par une commande left(90). Est-ce que tu devines avant de l'exécuter ce que sera le résultat ?

Puisque nous sommes dans le langage Python, je rappelle que la plage range(0,4) demande de répéter pour quatre éléments de zéro à trois et non de zéro à quatre. Tu obtiens bien les quatre côtés parce que tu commences à compter à zéro.

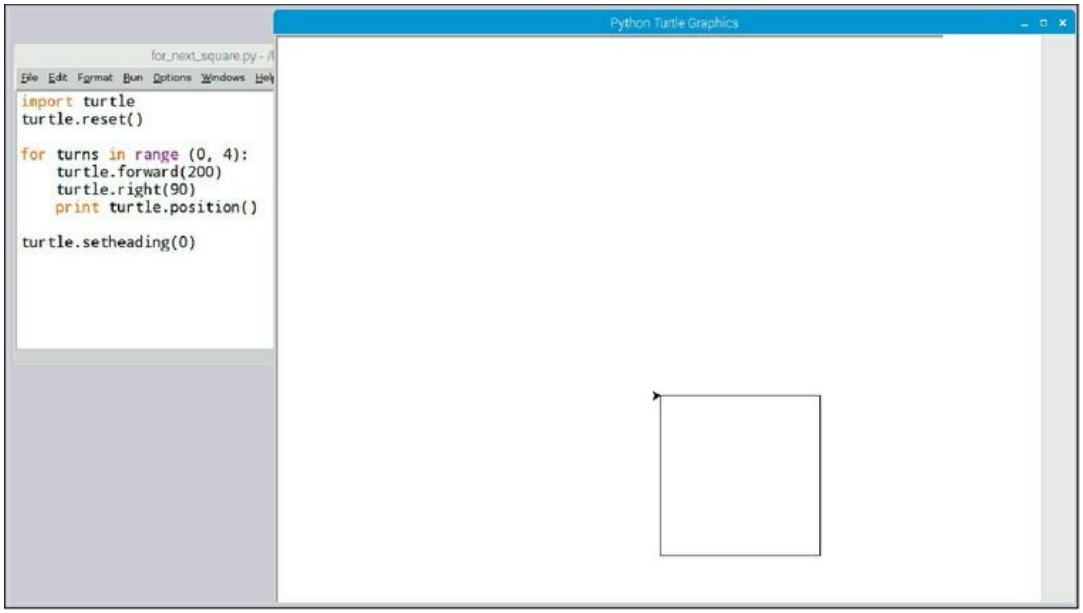


Figure 12.5 : Résultat de l'exemple for_next_square.py.

Où suis-je ?

Il est indispensable de savoir à tout moment où se trouve la tortue. Pour les dessins simples, ce n'est pas nécessaire, mais dès que tu vas effectuer des tracés un peu sophistiqués, tu ne pourras pas continuer si tu ne sais pas où tu te trouves, ni dans quelle orientation se trouve la tortue.

Dans la fenêtre graphique, la position est déterminée dans une grille de points quadrillée selon un axe horizontal et un axe vertical. Dans le sens horizontal, les valeurs vont de gauche à droite, et dans le sens vertical du bas vers le haut de la fenêtre.

Les coordonnées de chaque point dans la fenêtre correspondent à l'abscisse x pour le sens horizontal et à l'ordonnée y pour le sens vertical. Ces coordonnées sont en général indiquées entre parenthèses, comme ceci :

```
(200, -100)          # (x, y dans cet ordre)
```

Le centre de la fenêtre correspond à l'origine, qui a les coordonnées 0,0. Toutes les valeurs sont relatives à cette origine.

Dans d'autres systèmes, l'origine est dans le coin supérieur gauche de la fenêtre ou dans le coin inférieur gauche.

Ici, la tortue commence toujours au centre, ce qui rend obligatoire l'utilisation de coordonnées négatives vers la gauche et vers le bas.

La Figure 12.6 va t'aider à te repérer dans tes premiers pas. Si tu trouves ce dessin un peu trop mathématique, utilise le Tableau 12.2 qui complète l'illustration.

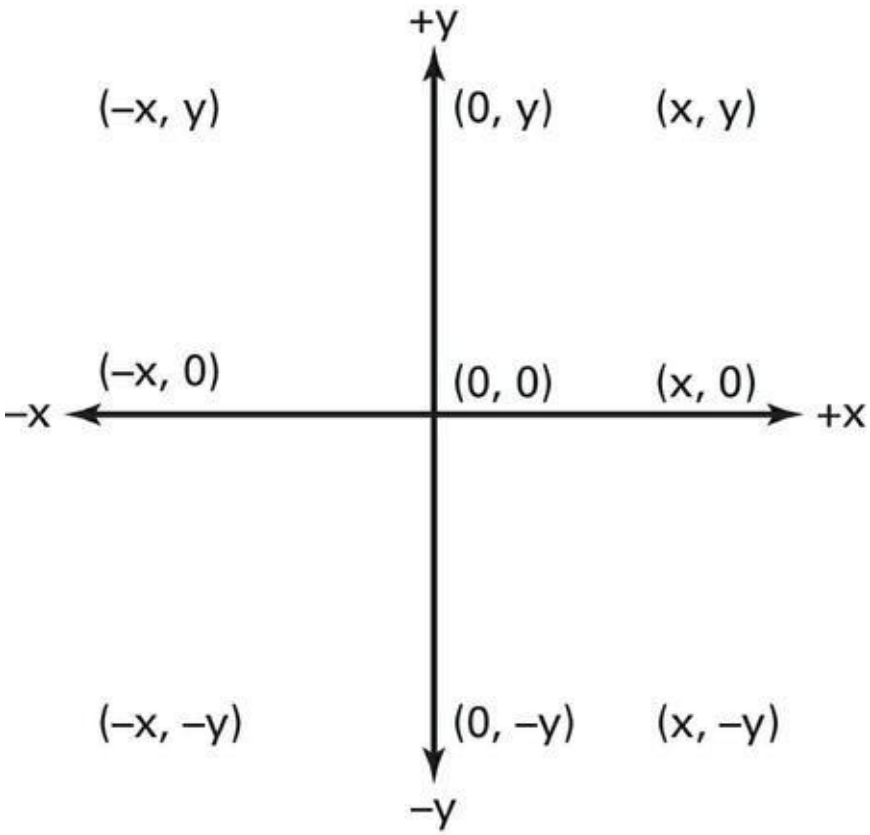


Figure 12.6 : Le système de coordonnées de la tortue.

Tableau 12 2 : Déplacements de la tortue.

Position	*Coordonnées*
À gauche du centre	-X en décroissant vers la gauche
À droite du centre	X, en augmentant vers la droite
Sous le centre	-Y, en décroissant vers le bas
Au-dessus du centre	Y, en augmentant vers le haut

Déplacements absolus

En utilisant le couple de coordonnées (x,y), tu peux positionner la tortue où tu veux dans la fenêtre. Tu disposes de la commande `setposition()` ou `setpos()` en version abrégée, ou bien la commande `goto()`. Dans les trois cas, il suffit d'indiquer le jeu de coordonnées. Ces trois lignes ont le même effet :

```
turtle.setposition(100, -100)
turtle.setpos(100, -100)
turtle.goto(100, 100)
```

N'hésite pas à envoyer ta tortue à différentes coordonnées jusqu'à ce que cette manière de se positionner te devienne évidente. Si tu es perdu, ajoute la commande `turtle.home()` pour ramener la tortue en plein milieu.

Pour ne déplacer la tortue que selon l'axe vertical ou horizontal, tu disposes de deux commandes :

```
turtle.setx(valeur)  # Translation
```

```
turtle.sety(valeur)  # Changement de hauteur
```

Tu peux même faire afficher les coordonnées actuelles. Les valeurs renvoyées par ces trois fonctions peuvent être réutilisées dans d'autres commandes afin de leur appliquer une opération arithmétique. Tu peux ensuite réutiliser le résultat avec des commandes de positionnement. Voici un exemple :

```
print turtle.xcor()      # Affiche la valeur
de x
print turtle.ycor()      # Affiche la valeur
de y
print turtle.position()  # Affiche (x, y)
```

Changer d'orientation

Pour orienter la tortue, tu disposes de la commande `setheading()`. Dans ce cas, tu n'utilises pas les valeurs d'angles relatives à l'orientation actuelle de la tortue, mais des directions absolues.

Autrement dit, l'orientation actuelle de la tortue n'a aucun effet pour cette commande. Son résultat est toujours la nouvelle orientation demandée.

Le Tableau 12.3 donne quelques exemples. Tu n'es bien sûr pas limité aux multiples de 90, sauf si tu ne veux dessiner que les carrés.

Tableau 12.3 : Angles de setheading().

Nouvelle rotation	*Orientation résultante*
0	droite
90	bas
180	gauche

270	haut

Tracer un cercle

Terminons cette partie sur les commandes de tracé avec la figure la plus évidente, le cercle. Voici un exemple :

```
turtle.circle(rayon)
```

La valeur détermine le rayon de ton cercle. Le cercle aura bien sûr un diamètre égal au double de la valeur fournie à la commande. Même pour une tortue, le diamètre d'un cercle est égal au double de son rayon.

Pour ne tracer qu'un arc de cercle, il faut indiquer une deuxième valeur après la virgule. Avec une valeur égale à 360, tu dessines un cercle complet. Pour un demi-cercle, tu indiques 180, et 90 pour un quart de cercle.

```
turtle.circle(rayon, arc)
```

Tu remarques que cette commande ne trace pas dans le sens des aiguilles d'une montre, mais dans le sens trigonométrique. Tu sais peut-être ce que sont les sinus et les cosinus ?

Contrôler la plume

Tu auras rapidement besoin de déplacer la tortue sans qu'elle fasse de trait. Autrement dit, il nous faut une commande pour lever la plume. Il s'agit de la commande `penup()` :

```
turtle.penup() # Levons la plume
# Ici, nous déplaçons la tortue
turtle.pendown() # Baissons la plume
```

Pour baisser la plume, tu utilises la commande `pendown()`. En jouant avec ces deux commandes, tu peux repositionner la tortue avant de commencer chaque nouveau tracé, sans laisser de traits de construction.

La vitesse de tracé

La vitesse standard de déplacement de la plume, pardon, de la tortue, permet de voir le tracé se faire, ce qui convient bien à des dessins simples. Lorsque tu vas demander des tracés plus élaborés, tu vas vite t'impatienter avant d'obtenir le résultat.

Tu peux très facilement augmenter la vitesse de la tortue avec deux lignes :

```
turtle.pen(speed = 0)
turtle.delay(0)
```

Le paramètre speed permet de régler une vitesse croissante entre 1 et 10. La valeur 0 est spéciale, car elle donne la vitesse maximale.

Tu peux également régler le délai entre deux actions avec la commande `delay()`. Si tu lui donnes la valeur 0, le programme ne fait plus aucune pause entre deux actions, ce qui convient bien à notre nano-ordinateur qui n'est pas très rapide.

Inversement, tu peux ralentir la tortue, afin de mieux distinguer par où elle passe.

Le délai entre deux actions correspond au temps de réaffichage de l'écran graphique après chaque opération. Lorsque ce délai est nul, Python réaffiche l'écran le plus vite possible.

La couleur de tracé

Pour contrôler la couleur du tracé, tu dois combiner les trois couleurs primaires rouge, vert et bleu. Les trois valeurs doivent être indiquées

en les séparant par des virgules.

Tu disposes de 256 valeurs différentes pour chaque couleur primaire, ce qui semble peu. Pourtant, 256 fois 256 fois 256 permet d'obtenir plus de 16 millions de nuances de couleurs, largement plus que ce que l'œil humain peut distinguer.

La tortue de Python accepte deux modes pour spécifier une nuance de couleur. Le premier mode consiste à indiquer une valeur entre 0 et 255 pour chaque composante. L'autre mode permet de définir la nuance avec une valeur entre 0 et 1, avec bien sûr des chiffres derrière la virgule, pardon derrière le point décimal.

En début de programme, tu choisis ton mode de couleur au moyen de la commande `colormode()`. Tu n'as besoin d'utiliser cette commande qu'une fois. Voici comment l'utiliser pour les deux modes :

```
turtle.colormode(1)    # Couleur entre 0 et 1

turtle.colormode(255)  # Couleur entre 0 et
255
```

Le mode de couleurs

Je rappelle que les ordinateurs commencent toujours à compter par zéro, c'est pourquoi la valeur maximale est égale à 255 et non 256. Il y a bien 256 valeurs différentes, car c'est le nombre de valeurs que l'on peut représenter avec 8 bits à zéro ou à un. Techniquement, il s'agit de deux à la puissance huit (2 x 2 x 2 x 2 x 2 x 2 x 2 x 2).

Voici comment spécifier une nuance avec le mode de couleur entre 0 et 255 :

```
255, 100, 0     # Beaucoup de rouge, moitié
de vert et pas de bleu
```

Voici comment spécifier des nuances avec le mode de couleur entre 0 et 1 :

```
(1, 0, 0) # Rouge vif
(0, 1, 0) # Vert
(0, 0, 1) # Bleu
```

Voici quelques nuances supplémentaires :

```
(1, 1, 1) # Blanc
(0, 0, 0) # Noir
(1, 1, 0) # Jaune
(1, 0, 1) # Magenta (violet clair)
(0, 1, 1) # Cyan (bleu clair)
```

Après le noir et le blanc, nous définissons ci-dessus les trois couleurs secondaires que sont le jaune, le magenta et le cyan. Je rappelle que ce sont les couleurs primaires pour les imprimeurs, c'est-à-dire en synthèse de couleur soustractive. Sur un écran, on utilise la synthèse additive.

Pour des nuances plus variées, il faut utiliser une valeur fractionnaire, comme dans les exemples suivants :

```
0.5, 0.5, 0.5      # Gris moyen
0.25, 0.25, 0.25   # Gris foncé
0.75, 0.75, 0.75   # Gris clair
```

Bien sûr, les valeurs des trois composantes peuvent être différentes pour obtenir encore d'autres teintes :

```
0.1, 0.2, 0.5     # Un bleu moyen
```

Tu peux même choisir des valeurs encore plus précises, mais tu vas arriver au bout des possibilités du système qui n'a que 256 valeurs différentes :

```
0.3333333, 0.3333333, 0.3333333     # Gris à
33%
```

Régler la couleur de plume

Puisque nous savons maintenant comment régler le mode de couleur et spécifier une nuance, nous pouvons choisir la couleur de tracé au moyen de la commande qui porte le nom `pencolor()`. Voici un exemple, qui correspond à la Figure 12.7 :

Listing 12.3 : Code source de turtle_circles.py

```
import turtle
turtle.reset()

turtle.colormode(255)
turtle.pencolor((0, 127, 255))

turtle.circle(100)
turtle.right(180)

turtle.pencolor((255, 127, 0))
turtle.circle(100)
```

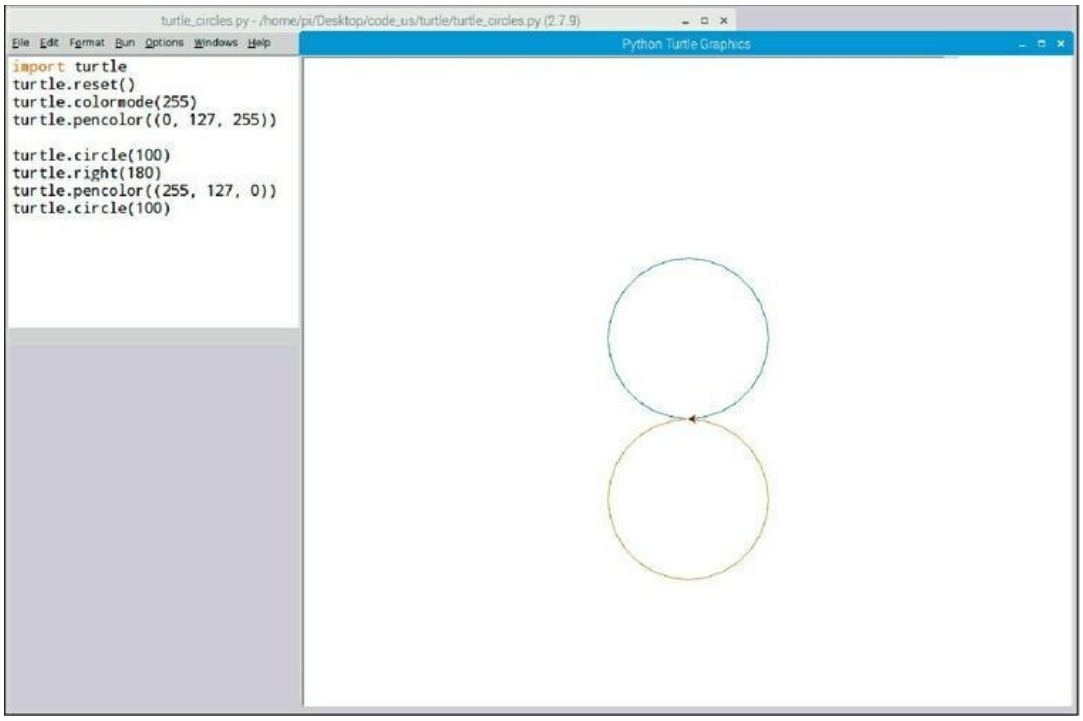


Figure 12.7 : Exécution de l'exemple turtle_circles.py.

Pour changer de couleur, tu utilises à nouveau la même commande.

Pour bien redémarrer

Parfois, tu auras besoin de repartir à zéro. Il suffit d'utiliser la commande suivante :

```
turtle.reset()
```

La commande ramène la tortue en plein milieu et orientée vers la droite.

Pour effacer tous les tracés actuellement présents, tu utilises la commande :

```
turtle.clear()
```

Cette commande efface l'écran, mais ne déplace pas la tortue. Elle est donc souvent combinée avec la précédente.

Créer une nouvelle fonction de tracé

En accumulant des commandes de déplacement et de rotation, tu peux réussir n'importe quel dessin, mais il te faudra bien de la patience pour réussir quelque chose d'un peu sophistiqué.

Tu as appris dans les chapitres précédents qu'un ordinateur permettait d'éviter les répétitions inutiles. Dès que tu dois répéter la même action, il faut chercher une technique pour simplifier l'écriture.

Simplifier le code source offre quatre autres avantages :

- Il y a moins à saisir.
- Il y a moins à réfléchir.
- Le risque d'erreur est moindre.
- Les possibilités sont bien plus grandes.

En programmation, tu peux aisément créer des blocs réutilisables. Une fois que tu as écrit un bloc permettant de tracer par exemple une maison, il suffit d'appeler ce bloc pour tracer autant d'exemplaires que nécessaire à différents endroits de l'écran.

Mais comment savoir quelles sont les commandes qu'il faut répéter ? Il suffit de chercher celles que tu as tendance à écrire plusieurs fois.

Il suffit de déplacer le code que tu dois répéter pour l'insérer dans le corps d'une nouvelle fonction. Nous avons déjà vu les fonctions au chapitre précédent ; elles dépendent du mot-clé `def`.

Si tu fais une erreur dans l'écriture de ce bloc, il n'y a qu'un endroit à retoucher.

Les fonctions que tu définis sont comme de nouvelles commandes qui s'ajoutent à celles disponibles dès le départ dans la librairie `turtle` de Python. Une fois qu'une fonction est définie, tu peux oublier les détails et t'en servir.

Définir une fonction

Pour commencer en douceur, voyons comment définir une fonction qui réalise un enchaînement de tracés.

Nous allons voir comment faire tracer une ligne puis tourner, qui sont les deux actions les plus utilisées dans Python Turtle.

Voici le code source qui doit être placé tout au début du programme pour définir la nouvelle fonction :

```
def tracer_tourner(distance, rotation):
    turtle.forward(distance)
    turtle.right(rotation)
```

Cette fonction devient immédiatement utilisable dans la suite du programme. Voici comment écrire un appel à cette nouvelle fonction avec de vraies valeurs :

```
tracer_tourner(200, 90)
```

Je rappelle que les fonctions que tu définis dans Python doivent être placées au début du fichier source, en tout cas avant le premier appel à chaque fonction.

Des constantes et des variables

L’utilisation la plus simple de notre nouvelle fonction pour tracer un carré par exemple ressemblerait à ceci :

```
tracer_tourner(200, 90)
tracer_tourner(200, 90)
tracer_tourner(200, 90)
tracer_tourner(200, 90)
```

Rien qu’en lisant ces quatre lignes, tu devines qu’il y a encore des répétitions inutiles. Il serait plus avisé de définir les variables une

fois, pour n'avoir qu'un endroit à modifier.

Tu dois te transformer en traqueur de répétitions inutiles. Voici comment définir deux variables puis les utiliser dans les appels à la fonction :

```
longueur = 200
angle = 90
tracer_tourner(longueur, angle)
tracer_tourner(longueur, angle)
tracer_tourner(longueur, angle)
tracer_tourner(longueur, angle)
```

Dorénavant, tu peux faire varier ton dessin en intervenant uniquement sur la valeur fournie au départ dans chacune des deux variables.

Tu peux même indiquer une valeur négative pour la longueur. Sais-tu à quoi ressemblera le résultat ?

Un bloc de répétition for

Dans l'état actuel de notre projet, nous avons quatre lignes qui répètent le même appel. On doit pouvoir optimiser cela. Voici comment avec une boucle `for` :

```
longueur = 200
angle = 90
for i in range (0, 4):
  tracer_tourner(longueur, angle)
```

Nous n'avons économisé que deux lignes, mais pour des figures plus complexes, la différence sera vraiment sensible.

Vers des tracés plus complexes

Si nous choisissons une valeur différente de 90 pour l'angle, nous n'allons plus tracer un carré, mais quelque chose d'autre.

En fait, si tu fais répéter un changement d'angle pas vraiment perpendiculaire, tu vas obtenir une figure bien plus créative, comme le montre la Figure 12.8.

Listing 12.4 : Code source de turtle_not_quite_square.py

```
def tracer_tourner(distance, rotation):
    turtle.forward(distance)
    turtle.right(rotation)

import turtle
turtle.reset()

longueur = 200
angle = 91

turtle.pen(speed = 0)
turtle.delay(0)

for i in range(0, 90):
    tracer_tourner(longueur, angle)
```

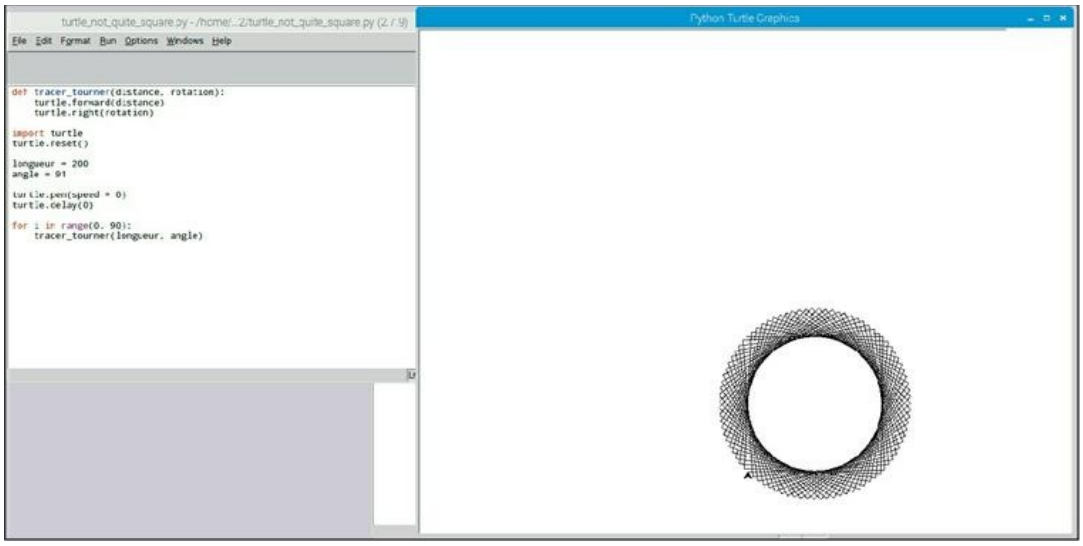


Figure 12. 8 : Figure produite par turtle_not_quite_square.py.

C'est ici qu'une boucle de répétition montre pleinement son avantage. Sans la boucle for, il aurait fallu écrire 90 lignes de tracé et 90 lignes de rotation !

Tu peux même rendre le résultat encore plus intéressant en ajoutant une ligne comme ci-dessous :

```
for i in range (0, 90):
  tracer_tourner(longueur, angle)
  longueur = longueur * 0.99
```

Nous avons ajouté une instruction pour diminuer la valeur de la variable `longueur` de 0.01 à chaque tour de boucle.

Teste cette version modifiée en ajoutant la ligne. Tu peux également essayer les actions suivantes :

- Tu peux changer légèrement la couleur de tracé à chaque tour de boucle.
- Tu peux varier les angles et sélectionner les résultats les plus intéressants ou curieux qui en résultent.

Se répéter soi-même ?

Dans l'exemple précédent, nous avions dès le départ décidé combien de tours de boucle nous allions faire. Il serait intéressant d'arrêter de dessiner dès qu'une condition est atteinte.

Il existe une technique assez sophistiquée pour répondre à ce problème. Elle consiste, accroche-toi, à appeler la fonction de l'intérieur d'elle-même, ce qui s'appelle la récursion.

Le principe de récursion

Le terme « récursion » vient de course. Une récursion consiste donc à parcourir une nouvelle fois un chemin déjà parcouru. Cela revient donc à répéter une ou plusieurs actions.

Une fonction n'est pas une boîte. Faire une récursion ne provoque pas l'insertion d'une boîte dans une autre. L'appel à la fonction de l'intérieur d'elle-même provoque le début d'exécution de la même fonction à partir de sa première instruction.

Pour résumer, sache que la récursion constitue une technique attrayante pour réaliser un certain nombre d'actions tant que la condition d'arrêt n'est pas rencontrée.

La récursion en pratique

Pour ajouter la récursion dans une fonction, il faut tenir compte de deux éléments :

- Il faut mettre en place un test qui permet d'arrêter d'appeler la fonction à un certain moment. S'il n'y a pas ce test, l'ordinateur va finir par se bloquer, car l'espace mémoire dans lequel sont stockées les adresses pour sortir des niveaux multiples de la fonction va déborder (erreur de pile d'appels).

> » L’autre élément est l’appel à la fonction de l’intérieur de cette fonction.

Parfois, il faut penser également à modifier dans la fonction la valeur d’une variable.

Voici la définition d’une fonction, proche de la précédente, mais avec les deux aménagements qui permettent d’entrer et surtout de sortir de la vrille que constitue la récursion :

```
def tracerSpirale(distance, rotation):
    if distance > 0:
        turtle.forward(distance)
        turtle.right(rotation)
        tracerSpirale(distance-5, rotation)
```

Tu constates que le corps de la fonction est sous la coupe d’une instruction conditionnelle if. Autrement dit, on sort de la fonction, et même de tous les étages successifs de la fonction dès que la longueur restant à tracer est nulle. La dernière ligne correspond à l’appel récursif à la fonction. Tu constates que le premier paramètre est réduit de cinq par rapport à sa valeur actuelle. Nous sommes ainsi certains qu’il finira par atteindre la valeur zéro, quelle que soit la valeur initiale. Le résultat visuel est que la ligne tracée est de plus en plus courte.

La Figure 12.9 montre le résultat avec une valeur de 121 pour l’angle de rotation. Ce code est assez intelligent puisqu’il s’arrête lorsqu’il n’a plus rien à tracer. Tu n’as pas besoin de prévoir le nombre de tours de boucle.

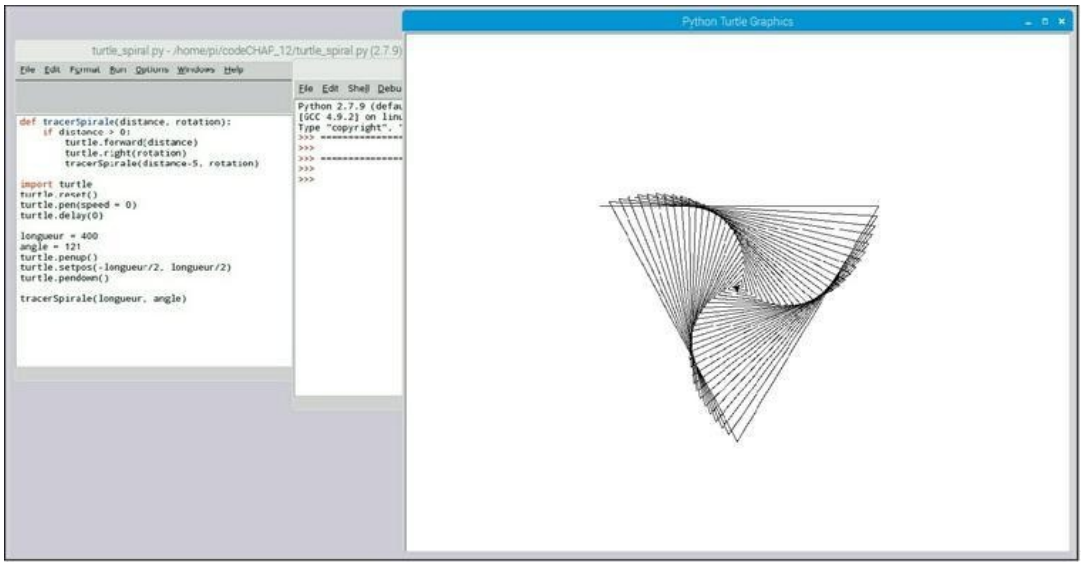


Figure 12.9 : Exécution de l'exemple turtle_spiral.py.

Listing 12.5 : Code source de turtle_spiral.py

```
def tracerSpirale(distance, rotation):
    if distance > 0:
        turtle.forward(distance)
        turtle.right(rotation)
        tracerSpirale(distance-5, rotation)

import turtle
turtle.reset()
turtle.pen(speed = 0)
turtle.delay(0)

longueur = 400
angle = 121
turtle.penup()
turtle.setpos(-longueur/2, longueur/2)
turtle.pendown()
```

```
tracerSpirale(longueur, angle)
```

Tout à la fin du code source, tu constates que le tracé est fait avec une seule instruction, l’appel à la fonction récursive.

La récursion est une excellente technique pour réaliser des tracés complexes, et notamment des fougères, des arborescences et des fractales. Tu trouveras dans les exemples disponibles sur le site de l’éditeur un exemple pour dessiner un arbre.

Chapitre 13
Du Minecraft dans Python

Tu sais sans doute que *Minecraft* est le plus utilisé des jeux vidéo de construction par blocs. Parallèlement, Python est l'un des langages de programmation les plus répandus. Ils sont donc faits pour s'entendre.

Découvrir Minecraft Pi

La version spéciale pour le Raspberry Pi de Minecraft est installée dès le départ. Tu la trouveras dans le sous-menu des jeux. Voici son écran d'accueil (Figure 13.1).

Figure 13.1 : L'écran d'accueil de Minecraft Pi.

Voici comment démarrer Minecraft :

1. Ouvre le menu général et choisis le sous-menu `Jeux` puis sélectionne `Minecraft Pi`.

   Tu constates que Minecraft s'affiche dans une fenêtre indépendante. Tu peux soit démarrer le jeu, soit rejoindre une partie en réseau.

2. Choisis l'option `Start game` puis demande la création d'un nouveau monde.

   Minecraft prend quelques instants pour construire un nouvel environnement dans la fenêtre de génération de monde.

La version Pi de Minecraft est allégée. Tu ne retrouveras pas toutes les options de la version complète : zones de danger, poulets agressifs, sorcières, ocelots, gelée collante et cochons volants. Pour tout savoir au sujet de Minecraft, rends-toi sur le site `http://minecraft.net`.

Découvrir le monde

Le monde de Minecraft est constitué de blocs en trois dimensions. La Figure 13.2 en donne un aperçu.

Avec la souris, tu peux déplacer ton point de visée. Pour te déplacer, tu utilises certaines touches du clavier, comme le montre le Tableau 13.1.

Figure 13.2 : Premier aperçu du monde de Minecraft.

Tableau 13.1 : Déplacements dans Minecraft.

Touche	*Déplacement*
W	En avant.
À	En crabe vers la gauche.
Des	En crabe vers la droite.
S	En arrière.

E	Affiche la fenêtre de choix des blocs.
Échap après E	Referme la fenêtre des blocs.
Espace	Un seul saut.
Espace 2 fois	Si au sol, commence à voler. Si en vol, retombe.
Tab	Libère la souris pour l'utiliser dans les autres fenêtres.
Échap	Revient au menu du jeu.

Pour changer de vue

Certaines personnes n'aiment pas le mode subjectif dans lequel tu vois avec les yeux du personnage. Le repérage n'est en effet pas toujours facile.

Tu peux changer de vue en frappant la touche Echap puis en cliquant dans le deuxième bouton dans le coin supérieur gauche de la fenêtre. Ce bouton doit montrer un rectangle avec des traits qui mènent vers deux blocs. Tu peux alors frapper Echap ou choisir `Back to game`.

À partir de ce moment, tu vois le joueur de dos. Certaines personnes préfèrent jouer de cette façon pour mieux voir ce qu'il y a autour du joueur.

Intervenir sur le monde

Le but de Minecraft est de donner toute liberté pour créer et modifier le monde dans lequel tu es placé. Pour enlever un bloc, il suffit de

cliquer avec le bouton gauche afin d'utiliser ton épée. Après plusieurs coups, le bloc se brise et disparaît.

Ce n'est pas très simple de bien viser dans Minecraft. Il faut s'entraîner un minimum pour savoir où va porter ton prochain coup.

Si rien ne se passe quand tu essaies de frapper, déplace ton joueur jusqu'à voir un bloc devant toi.

Pour ajouter des blocs, utilise la touche E pour afficher ton stock de blocs et d'autres objets. Clique un objet pour le sélectionner. Si c'est un bloc de construction, tu peux ensuite cliquer-droit pour l'ajouter dans ton monde. Il suffit de répéter l'opération pour finir par construire quelque chose, comme le montre la Figure 13.3.

Figure 13.3 : Un joueur en pleine construction.

Une API pour contrôler une appli

API est un acronyme qui signifie Application Programme Interface. C'est le nom que l'on donne à un ensemble de fonctions que tu peux appeler dans tes programmes pour contrôler le comportement d'un

autre programme, d'un site Web, d'un jeu vidéo ou d'une autre application. Tu déclenches des événements dans cet autre programme non plus en cliquant ou en tapant au clavier, mais en envoyant des commandes logicielles qui sont écrites dans le code source. L'interface API permet par exemple de récupérer des informations sur le contenu du site, du jeu ou de l'application.

Il y a des interfaces API pour quasiment tous les programmes. Par exemple, Twitter, Facebook et tous les grands sites Web disposent d'une API. Voici quelques-unes des choses que tu peux réaliser en utilisant l'interface API de Twitter. La liste complète des possibilités est bien plus grande.

- Déclencher automatiquement certaines choses. Avec l'API de Twitter tu peux faire émettre automatiquement des tweets à certaines heures, même lorsque tu n'es pas connecté.
- Collecter des informations. L'API permet de demander à Twitter une fois par jour combien de personnes suivent ton compte, pour produire ensuite un graphique.
- Ajouter des fonctions intéressantes. Tu peux par exemple vouloir suivre toute personne qui publie un mot en particulier, comme le nom d'un groupe musical ou le nom d'une actualité récente. Tout cela est possible grâce à l'interface API.

L'interface API de Minecraft

L'API de Minecraft, qui est disponible dans la version Pi, est plus simple que celle de Twitter. Elle permet néanmoins de faire des choses extraordinaires, dont voici une courte sélection :

- Afficher les coordonnées actuelles du joueur.
- Se téléporter.
- Construire automatiquement des bâtiments complexes.
- Supprimer des blocs, par exemple pour libérer une zone.

Informations sur l'API Minecraft

Puisqu'une interface API est d'abord une liste de noms de fonctions et de règles d'utilisation, tu as nécessairement besoin d'un site de référence qui liste tous les appels API.

La référence de l'API Minecraft en anglais se trouve à l'adresse suivante :

www.stuffaboutcode.com/p/minecraft-api-reference.html

Il existe également une petite introduction à l'API en français à l'adresse suivante :

www.minecraft.fr

Figure 13.4 : Page d'accès aux informations sur l'API Minecraft en français.

Naviguer parmi les pages de la référence n'est pas toujours simple parce que les noms des fonctions que tu peux appeler sont mentionnés, mais les explications sont souvent trop brèves, et les exemples peu nombreux.

Heureusement, l'API de Minecraft possède quelques bons exemples. Attends-toi cependant à travailler par essais et erreurs.

Une dernière remarque : tous les appels sont considérés comme ayant la même importance, alors que certains seront très souvent utilisés et d'autre presque jamais.

Une bonne technique consiste à copier/coller la description de la fonction dont tu as besoin pour pouvoir l'étudier tranquillement et faire des essais. Tu peux ensuite chercher d'autres exemples sur le Web.

Ne tente pas d'apprendre tous les appels de l'API avant de commencer à l'utiliser. Cela risquerait de te prendre la tête. Il vaut mieux picorer en fonction de tes besoins. Au bout d'un certain temps,

tu sauras par cœur les noms des appels très utilisés.

Utiliser l'API Minecraft

Dans un programme qui va utiliser une interface API, il y a normalement toujours un embryon de code au début du programme pour préparer le dialogue avec l'API. Dans le cas de Minecraft, ce code initial correspond aux deux lignes suivantes :

```
from mcpi import minecraft
mc = minecraft.Minecraft.create()
```

La première ligne demande d'importer la librairie mentionnée pour qu'elle soit utilisable dans ton programme. Dans la deuxième ligne, nous créons une variable portant le nom `mc`. Cette variable va incarner ton joueur, sous forme d'un robot Minecraft. Tu vas ensuite envoyer des commandes à `mc` (en réalité, c'est un objet) qui va les retransmettre à Minecraft qui va réaliser l'action demandée ou renvoyer les données réclamées.

Le principe d'une API consiste à envoyer des messages qui contiennent des commandes. Cela permet de masquer les détails techniques. Tu peux ainsi rester concentré sur les grandes lignes de ce que tu veux réaliser.

Notre premier appel API

Commençons par un exemple simple d'utilisation de l'API :

1. Ouvre le menu principal, le sous-menu `Programmation` et lance l'éditeur de la version 2 de Python, `Python 2 IDLE`.
2. Dans la fenêtre, démarre un nouveau projet en ouvrant le menu `File` puis en choisissant `New file`.

3. Dans la fenêtre d'édition, saisis les quatre lignes de code suivantes :

Listing 13.1 : Code source de mcPosition.py

```
from mcpi import minecraft
mc = minecraft.Minecraft.create()

x, y, z = mc.player.getPos()
print x, y, z
```

Tu devines peut-être déjà quel sera le résultat.

4. Enregistre ce fichier sous le nom mcPosition.py.
5. S'il n'est pas encore ouvert, démarre Minecraft.
6. Effectue quelques déplacements dans le jeu.
7. Accède alors à la fenêtre de code dans l'éditeur Python, par exemple avec la combinaison de touches Alt + Tab. Frappe alors la touche F5 pour lancer l'exécution.
8. Effectue à nouveau un déplacement dans le jeu puis relance l'exécution du programme.

   Tu dois obtenir plusieurs affichages de coordonnées, comme dans la Figure 13.5.

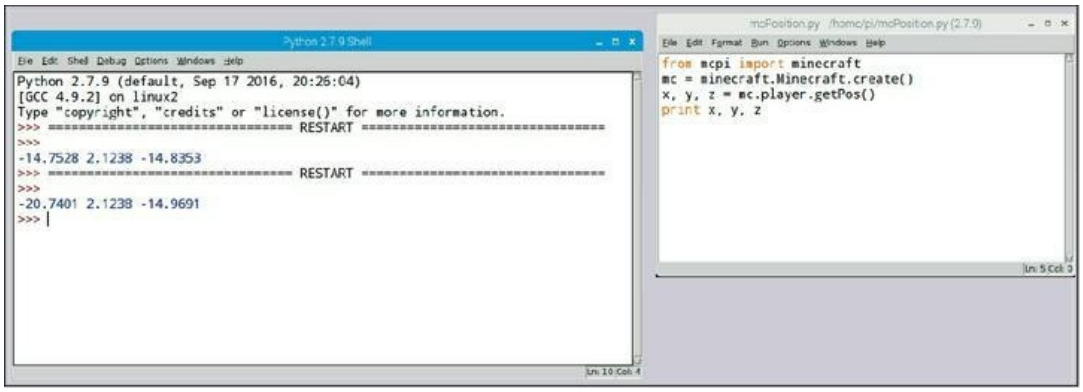


Figure 13.5 : Exemple d'affichage de coordonnées de Minecraft.

Ce programme indique donc la position du joueur dans le monde Minecraft avec trois coordonnées qui sont expliquées dans le Tableau 13.2.

Tableau 13.2 : Coordonnées dans Minecraft.

Lettre	*Description*
x	Vers l'est et l'ouest.
y	Vers le haut et le bas pour voler et creuser.
z	Vers le nord et le sud.

Il n'y a pas de boussole dans Minecraft. Les valeurs de x et de z servent donc à se repérer de façon plus ou moins virtuelle. Le nord ne change jamais d'orientation.

Se téléporter dans Minecraft

Il existe un appel API pour se déplacer à la vitesse de la lumière dans Minecraft.

1. Tu peux repartir du code source précédent. Modifie la quatrième ligne, comme le montre l'exemple suivant.

Listing 13.2 : Code source de mcSauter.py

```
from mcpi import minecraft
mc = minecraft.Minecraft.create()
x, y, z = mc.player.getPos()
mc.player.setPos(x, y+100, z)
```

Nous utilisons ici la fonction `setPos()` pour changer les coordonnées du joueur, et donc le déplacer.

2. Tu peux te déplacer un peu pour l'instant.

   Tu peux travailler avec des coordonnées numériques relatives à ta position actuelle pour décider de la grandeur du saut. Tu peux même fournir des nombres tirés au sort pour retomber n'importe où.

3. Enregistre ce nouveau projet sous le nom `mcSauter.py` puis lance l'exécution par F5.

Le résultat ressemble au contenu de la Figure 13.6. Avec les valeurs de l'exemple, le joueur remonte de 200 blocs d'un coup.

Figure 13.6 : Exemple d'exécution de mcSauter.py.

Je rappelle que tu ne meurs pas lorsque tu t'écrases, car le joueur est constitué de blocs très solides.

Détruire des blocs

Parfois, l'idée de devoir détruire un grand nombre de blocs à coups d'épée est vraiment décourageante. Il y a sûrement une technique plus rapide.

Si tu cherches dans la référence de l'API, tu ne trouveras aucun bloc qui permettrait directement de supprimer un bloc, du style `deleteBlock()`. En revanche, il existe une fonction pour changer la matière d'un bloc, elle s'appelle `setBlock()`.

Pour être exact, on ne détruit jamais de bloc dans Minecraft. La totalité du monde est constituée de blocs qui se touchent les uns les autres, même dans l'air. Il y a donc une astuce.

Là où tu ne vois pas de blocs, ce sont en fait des blocs dans la matière Air. Voilà pourquoi tu peux utiliser cette fonction pour convertir un

bloc de pierre en bloc d’air, ce qui le fait disparaître.

Voici un exemple qui le prouve.

Listing 13.3 : Code source de mcCreuser.py

```
from mcpi import minecraft
from time import sleep

mc = minecraft.Minecraft.create()
while True:
    x, y, z = mc.player.getPos()
    mc.player.setPos(x, y-1, z, 0)
    sleep(0.1)
```

Comme son nom l’indique, le jeu Minecraft permet de creuser une mine. Dans cet exemple, le bloc sous les pieds du joueur est détruit en rendant aérien le bloc sous les pieds du joueur. Il y a une boucle éternelle, ce qui te fait descendre de plus en plus.

Saisis le code source de ce programme et enregistre-le sous le nom `mcCreuser.py`. Lance ensuite l’exécution. S’il ne se passe rien, déplace un peu ton joueur. Tu finiras par voir apparaître le fatidique écran de la Figure 13.7.

En effet, puisque tu creuses sans cesse, il arrive un moment où Minecraft considère que tu as atteint le fond et il provoque la mort du joueur.

Figure 13.7 : Stop ou encore ?

Toute l'astuce réside dans le quatrième paramètre de la fonction `setBlock()`. La valeur zéro demande d'utiliser la matière Air. Minecraft offre des centaines de matières différentes. Tu peux en retrouver une liste dans la référence API, avec le numéro de chacune.

Et maintenant, construisons !

Tu peux bien sûr faire l'inverse : transformer un bloc d'air en bloc de pierre ou d'eau. Essaie le projet suivant.

Listing 13.4 : Code source de mcConstruire.py

```
from mcpi import minecraft
import random

mc = minecraft.Minecraft.create()
x, y, z = mc.player.getPos()
```

```
x = x + random.randint(-10, 10)
z = z + random.randint(-10, 10)

for i in range(0, 21):
    mc.setBlock(x, y+i, z,
random.randint(1,8))
```

Étudie le code source pour essayer de deviner ce que sera son résultat. L'appel de fonction `randint()` sert à générer une valeur numérique aléatoire située dans la plage définie par les deux nombres. Par exemple :

```
random.randint(-10, 10)
```

Cela génère à chaque appel un nombre entre -10 et +10. Le nombre change à chaque appel, de manière normalement imprévisible.

Les nombres aléatoires permettent d'ajouter de l'intérêt à un jeu. Tu définis les règles générales dans le code source, et la fonction ajoute des variations de détail qui ne sont jamais les mêmes.

Dans notre exemple, nous choisissons au hasard les coordonnées x et z, de sorte qu'elles restent proches du joueur, mais pas exactement au même endroit.

Ce positionnement permet de construire ensuite le bâtiment en convertissant des blocs Air en autres blocs.

Nous utilisons la même fonction de génération aléatoire pour décider de la matière des blocs. Avec une plage entre 1 et 8, certains des blocs seront une cascade. La Figure 13.8 montre un résultat possible : une fontaine !

Figure 13.8 : Exécution du projet mcConstruire.py.

L'eau va s'écouler de la fontaine jusqu'au sol puis jusqu'à la mer. Lors de chaque exécution, la fontaine aura un aspect légèrement différent.

Pour aller plus loin

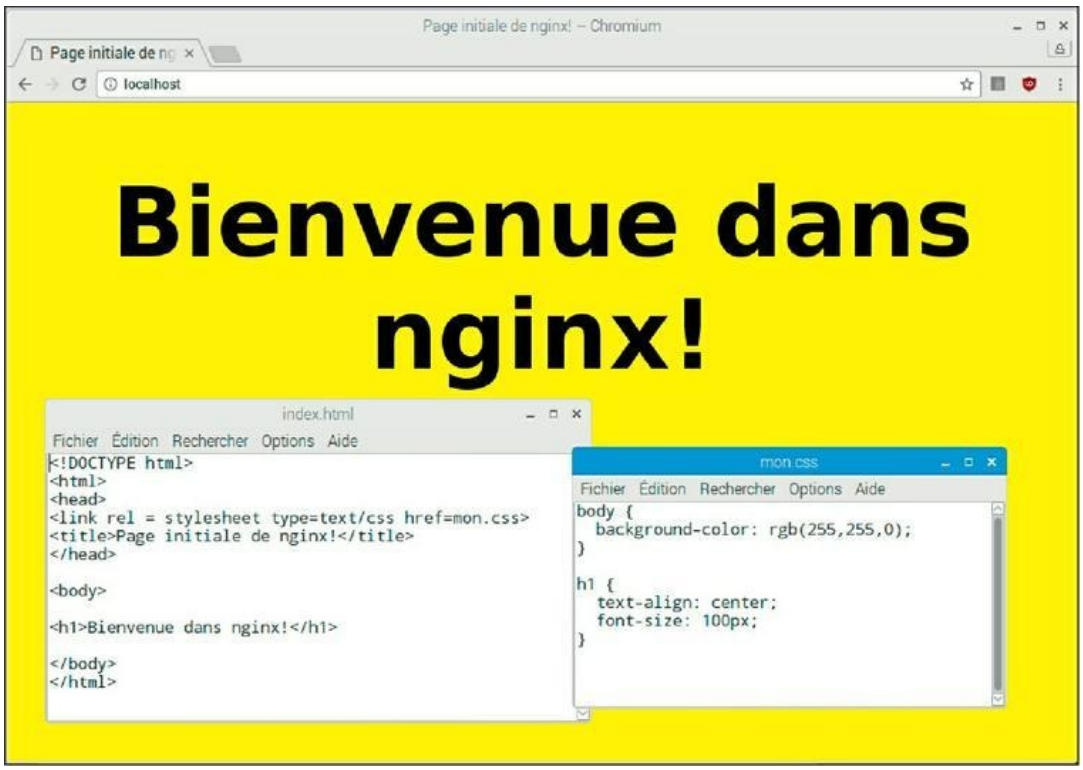
Je termine avec quelques idées pour poursuivre. Certains de ces projets sont plus difficiles à réaliser que d'autres. N'hésite pas à tous les essayer. Tu peux prévoir qu'il faudra faire des recherches et en apprendre plus sur l'utilisation du langage Python.

» Supprimer des blocs en marchant. Ce projet n'est pas aussi simple qu'il semble. Combien faut-il enlever de couches de blocs avant de pouvoir marcher dans le trou ? Quelle est la largeur à prévoir ? La solution évidente consiste à répéter les appels à `setBlock()`. Il y a peut-être une technique plus efficace en utilisant des boucles et des tests ?

- Construire de grands modules. Construire un cube est assez facile. Il suffit d'empiler des blocs. Mais saurais-tu construire une pyramide ou une série de passerelles ? Tu peux même essayer de faire des formes circulaires ou une véritable maison.
- Construire une fusée. Tu peux partir d'une tour ou de la fontaine et la faire voler. Mais comment faire en sorte que les blocs restent collés ensemble ?
- Créer un feu d'artifice. Une fois que tu as fait la fusée, amène-la par exemple à 100 blocs de hauteur puis fais exploser tous les blocs dans des directions différentes. Comment faire pour conserver la position de chaque bloc ?
- Construire un labyrinthe. Ce dernier projet ne sera vraiment pas facile, à moins de faire un labyrinthe trop simple. N'hésite pas à consulter sur Wikipédia la page intitulée « Modélisation mathématique d'un labyrinthe ». Essaie ensuite de trouver sur le Web un exemple de code Python et adapte-le pour qu'il soit utilisable avec Minecraft. Les murs doivent être suffisamment hauts pour que le joueur ne puisse pas s'en échapper.

Chapitre 14
Un site Web minimal

Ton Raspberry Pi ne coûte peut-être que quelques dizaines d'euros, mais tu peux très facilement t'en servir pour configurer un véritable serveur Web. Nous allons commencer par une page Web extrêmement simple, mais tu pourras y accéder depuis n'importe quel autre ordinateur de ton réseau.



Le principe des serveurs Web

Un serveur Web est un logiciel technique. Son rôle est de se tenir à l'écoute de demandes d'affichage de pages Web pour y répondre en renvoyant les données de ces pages.

Les sites Web qui ont une forte fréquentation, comme Facebook (Figure 14.1), doivent répondre jour et nuit à des centaines de milliers de requêtes provenant des navigateurs des visiteurs. C'est la raison pour laquelle ces serveurs Web sont dédiés à cette fonction. Ce serait tout de même dommage de voir le temps de réponse de Facebook s'écrouler parce qu'un technicien du centre de calcul a décidé de jouer à Minecraft sur un serveur Web. Dès que la fréquentation d'un site commence à croître, l'application qui incarne le serveur Web doit être quasiment la seule à fonctionner sur l'ordinateur.

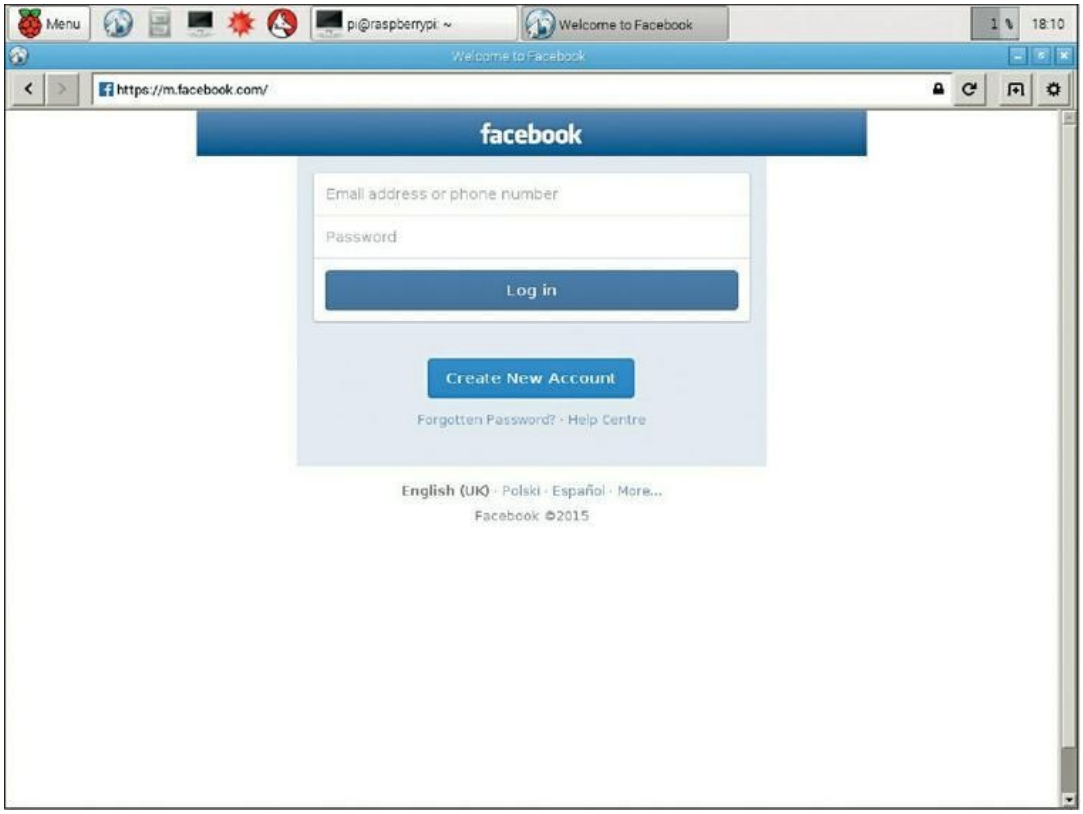


Figure 14.1 : Page d'accueil du site Facebook.

Bien sûr, ton nano-ordinateur est loin d'offrir assez de puissance pour y loger un serveur Web ayant une certaine fréquentation, et sans compter qu'un grand site Web exploite des outils complémentaires qui ont été développés uniquement dans ce but. Ces outils logiciels ne sont pas disponibles gratuitement.

En revanche, le serveur Web que tu vas installer dans ce chapitre pourra répondre à cinq ou dix visiteurs simultanés.

Un site Web statique

On dit qu'un site Web est statique dès lors qu'il ne fait que renvoyer des pages dont le contenu est toujours identique. La quasi-totalité du contenu disponible sur le Web est effectivement constituée de mots, c'est-à-dire de texte, avec quelques images et des liens. Parfois, tu trouves également des animations, quelques menus et autres améliorations.

Le contenu d'un site Web statique n'est modifiable que si l'auteur de la page la charge dans un éditeur de texte puis télécharge la nouvelle version dans l'espace du site Web. Si tu ne changes pas ce contenu, l'aspect du site sera toujours le même, avec les mêmes textes, les mêmes images et les mêmes liens.

Choix d'un serveur Web

Comme tous les autres programmes sur le Raspi, tu vas installer un serveur Web au moyen de l'outil universel `apt-get`. Puisque nous sommes sous Linux, tu as un vaste choix de programmes serveurs, dont le Tableau 14.1 donne une sélection.

Dans ce chapitre, je te propose d'opter pour le logiciel serveur portant le nom nginx (engine X). Il est de bonne qualité et facile à mettre en place.

Tableau 14.1 : Quelques serveurs Web pour le Raspi.

Non	*Domaine d'emploi*
Apache	Serveur Web professionnel très complet, mais pas très simple à administrer.
nginx	Serveur Web complet, plus simple à configurer qu'Apache.

lighttpd	Serveur Web simplifié, facile à configurer, mais ne convenant qu'aux projets les plus simples.
node.js	Serveur Web apparu récemment. D'une grande souplesse, mais trop complexe pour un débutant.

Ne tente jamais d'installer deux serveurs à la fois. Lorsque tu auras envie d'essayer Apache à la place de nginx, mieux vaut repartir d'une nouvelle carte mémoire avec une réinstallation de Raspbian. Les serveurs Web n'aiment pas se faire concurrence.

Installer nginx

L'installation de ce serveur est très simple :

1. Depuis le bureau graphique, ouvre une fenêtre de terminal. Tu dois voir apparaître l'invite de ligne de commande. Revois les Chapitres 5 et 9 pour en savoir plus au sujet de la fenêtre de terminal et de la ligne de commande.
2. Saisis la commande d'installation suivante, puis valide par entrée :

```
sudo apt-get install nginx
```

L'installation se déroule comme le montre la Figure 14.2. Il est possible qu'une demande de confirmation apparaisse. Tape la lettre Y ou O majuscule. Il suffit d'attendre que l'invite de commande réapparaisse.

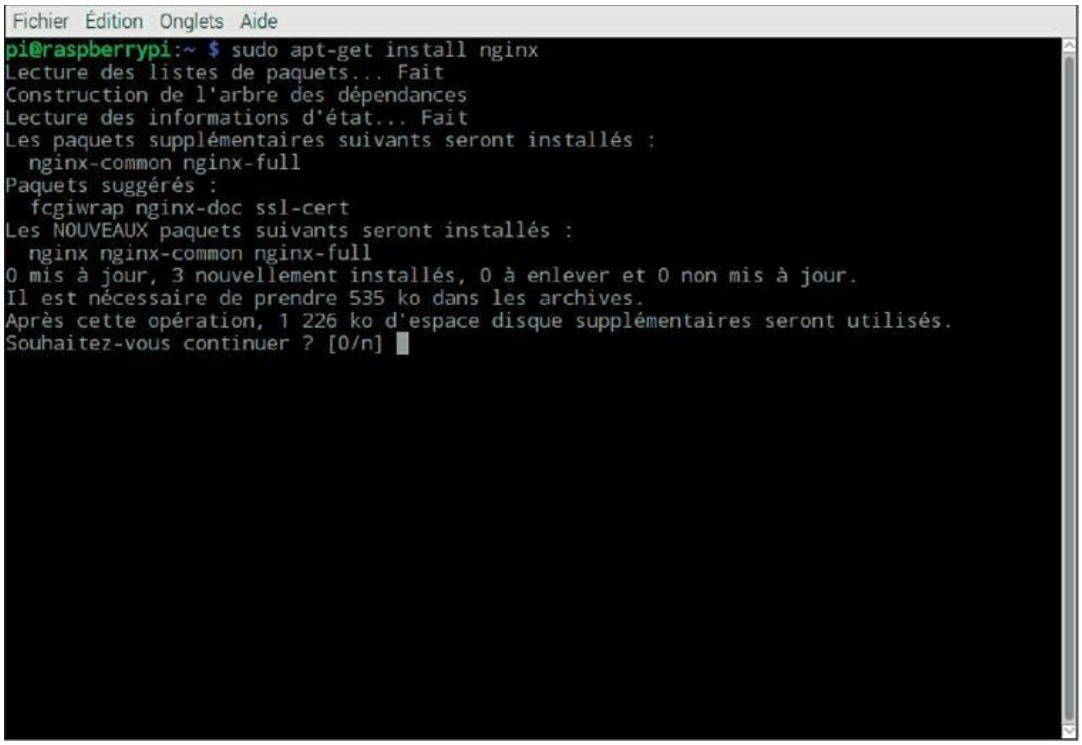


Figure 14.2 : Installation du serveur Web.

Démarrer le serveur Web

Dès que l'installation est achevée, tu peux démarrer ton serveur, ce qui suppose pour simplifier de redémarrer l'ordinateur.

Puisque la fenêtre de terminal est toujours ouverte, il suffit de saisir la commande suivante et de valider par Entrée :

```
sudo reboot
```

Dans l'interface graphique, il suffit d'ouvrir le menu général puis de choisir `Shutdown`. Clique alors le bouton `Reboot` puis OK.

Lorsque la machine a redémarré, ton serveur est en fonctionnement, même si cela ne se voit pas.

Tester le serveur Web

Voici comment faire pour vérifier que le serveur nginx est effectivement prêt à répondre à nos demandes :

1. Ouvre le navigateur Web et clique dans la barre d'adresse puis saisis ceci :

   ```
   localhost
   ```

   Cette adresse spéciale demande d'accéder au serveur Web qui se trouve sur le même ordinateur que le navigateur.

La Figure 14.3 montre la page que tu devrais voir apparaître, ou une page très similaire. Il s'agit vraiment d'une page qui a été transmise par le serveur Web.

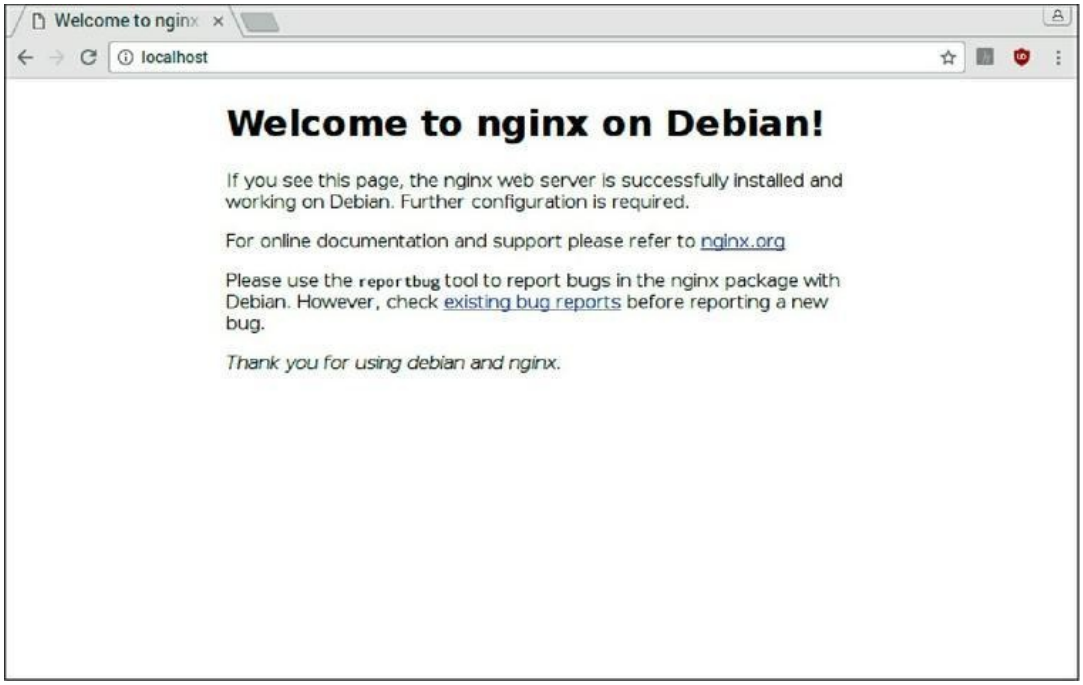


Figure 14.3 : Affichage de la page d'accueil lors du test du serveur Web.

Nous n'en avons pas terminé avec les tests.

2. Si tu connais déjà l'adresse IP de ton nano-ordinateur (j'en ai parlé dans le Chapitre 5), installe-toi devant un autre ordinateur du même réseau local, ouvre le navigateur et saisis cette adresse IP dans la barre d'adresse.

3. Si tu ne connais pas l'adresse IP de ta machine, ouvre une fenêtre de terminal sur le Raspi et saisis la ligne de commande suivante (c'est un I majuscule) :

```
hostname -I
```

4. Dès que tu valides par Entrée, tu vois s'afficher ton adresse IP.

Si tu as bien saisi l'adresse, tu dois pouvoir afficher la page Web du serveur depuis n'importe quel autre ordinateur, tablette ou téléphone branché sur le même réseau local.

Voyons maintenant comment modifier cette page de test, afin de prendre possession du contenu que nous publions.

Ton serveur Web peut être rendu accessible depuis tout l'univers, à partir du moment où tu loues un nom de domaine, comme par exemple `monnom. com`. Cela dit, les choses ne sont pas très simples pour arriver à ce but. Faire en sorte que le serveur Web du Raspi devienne un vrai serveur Web public est une grande aventure, d'autant qu'il faut particulièrement soigner les aspects de sécurité.

Créer une page Web simple

La page de test n'est pas très intéressante, mais avant de nous en servir pour créer une première page personnalisée, voyons d'abord les différentes opérations. Tout d'abord, où sont stockés les fichiers de contenu du serveur ? Pour la plupart des logiciels de serveurs Web,

il y a un chemin d'accès standardisé. Dans le cas de nginx, le chemin est normalement le suivant :

```
/var/www/html
```

Pour le vérifier, ouvre le Gestionnaire de fichiers de l'interface graphique et parcours les sous-répertoires jusqu'à celui indiqué ci-dessus. Tu dois trouver au moins un fichier dont l'extension est `.html`, ce qui signifie que c'est un fichier de contenu Web. Ce fichier contient le texte qui est affiché, ainsi que des instructions qui contrôlent la manière dont ce texte apparaît, c'est-à-dire des balises.

HTML signifie *Hyper Text Markup Language*, c'est-à-dire langage de balisage hypertexte. C'est le langage dans lequel est écrit le contenu d'un fichier à l'extension `.html`. (Tu rencontreras également parfois des fichiers identiques mais dont l'extension est `.htm`.)

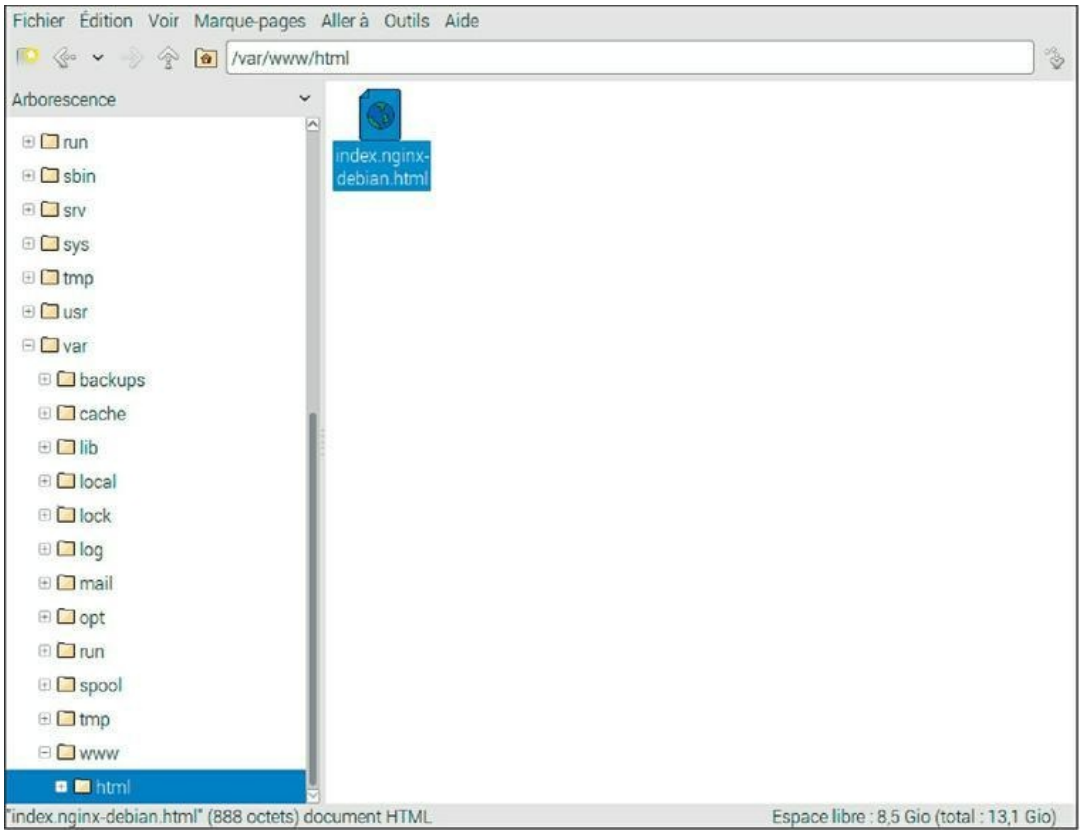


Figure 14.4 : Le sous-répertoire contenant les pages Web.

Découvrir le fichier index.html

Lorsque tu saisis dans ton navigateur une adresse sans spécifier le nom d'une page, le serveur va tenter de te renvoyer la page standard qui porte le nom `index.html`.

Certaines versions de nginx proposent un fichier nommé `index.nginx-debian.html`. Tu peux le renommer en `index.html`.

Autrement dit, pour modifier le contenu de la page de test, nous allons intervenir sur le contenu du fichier qui porte ce nom.

Puisque tu es toujours dans le Gestionnaire de fichiers, clique-droit dans le nom ou l'icône du fichier `index.html` et choisis la commande `Text Editor`. Le fichier est chargé dans l'éditeur de texte (Figure 14.5).

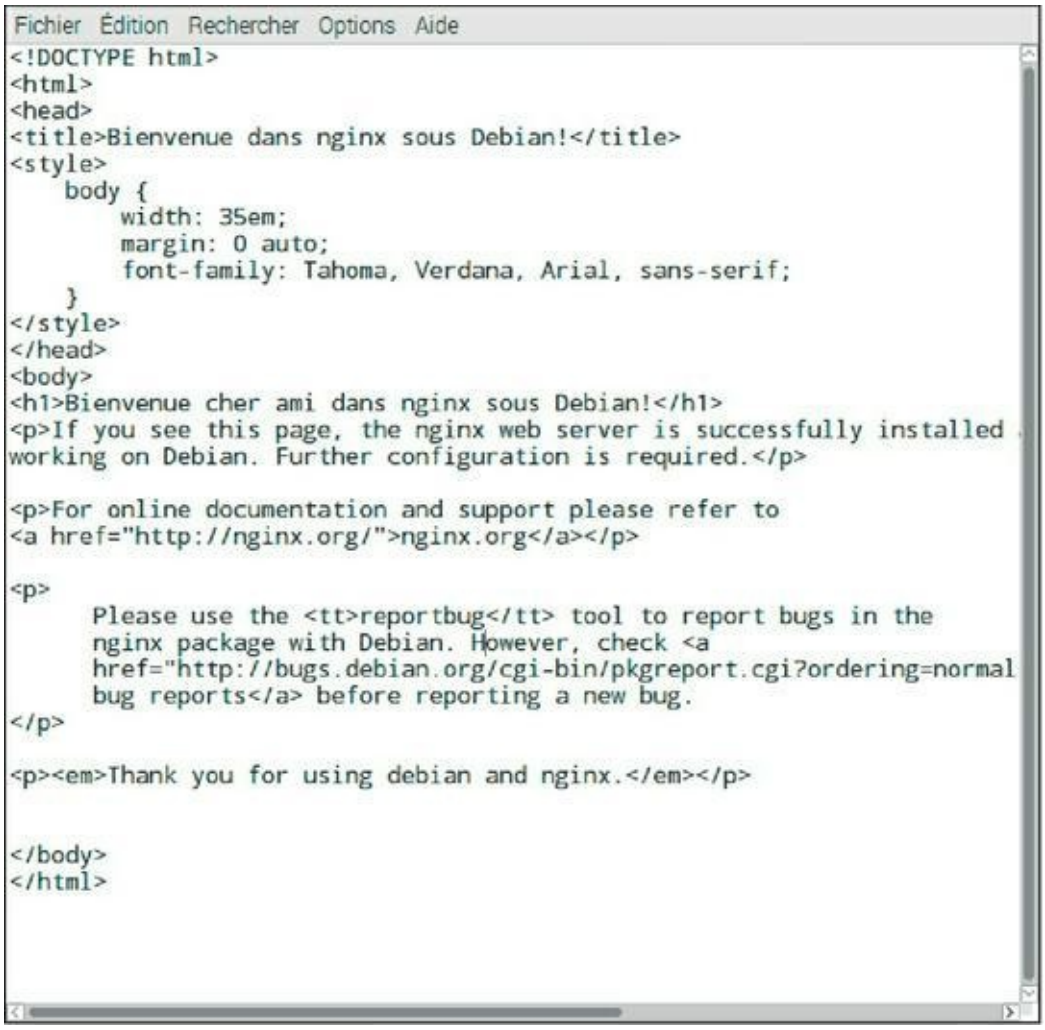


Figure 14.5 : Édition d'un fichier HTML dans l'éditeur.

Dans certaines versions du serveur nginx, tu vas trouver un autre fichier dans le même répertoire, portant le nom `50x.html`. C'est le fichier qui est affiché lorsque la page demandée est introuvable, c'est-à-dire un fichier pour l'erreur numéro 404. Pour une raison inconnue, les créateurs du serveur ont décidé d'appeler ce fichier `50x`.

Principe des balises HTML

Lorsque tu observes le contenu du fichier HTML, tu constates que c'est différent d'un fichier de code source Python. Tu trouves beaucoup de mots encadrés par des <chevrons>.

Tous ces mots encadrés sont des *balises* ; les balises ne sont jamais visibles dans le navigateur. Le principe d'une page HTML est de comporter plusieurs sections, au minimum deux: une pour les paramètres techniques et une pour le contenu. Tu peux définir d'autres sections selon tes besoins.

Chaque section est délimitée par un couple de balises, une balise ouvrante (de début) et une balise fermante (de fin). La balise fermante se distingue de la balise ouvrante par la présence d'une barre oblique entre le chevron ouvrant et le nom. Voici un exemple :

```
<title>Ceci est le titre</title>
```

Cette ligne définit une section de titre, les deux balises délimitant le contenu.

Le contenu de cette section `<title>` ne sera pas affiché dans la fenêtre du navigateur, mais dans la barre de titre du navigateur.

Les balises <html>, <head> et <body>

Un certain nombre de balises servent à préparer l'affichage de la page par le navigateur. Elles sont réunies dans la première section qui porte le nom `<head>`, c'est-à-dire *en-tête*.

Les autres balises vont constituer le contenu, c'est-à-dire le *corps de la page*. La balise porte donc le nom `<body>`.

Normalement, il faut également ajouter une balise `<html>` qui englobe toutes les autres, car le standard autorise d'autres langages proches du HTML, comme le XML et le SGML.

Voici l'aspect général d'une page Web minimale :

Listing 14.1 : Structure générale d'une page Web.

```
<html>
  <head>
    Paramètres de préparation de la page...
  </head>
  <body>
    Contenu de la page avec d'autres
balises...
  </body>
</html>
```

Les indentations ne sont pas obligatoires ; je les ai ajoutées pour faciliter la compréhension. Ces espaces sont ignorées dans l'affichage.

Les pages Web normalisées contiennent également une définition de type de document avec la balise `< ! DOCTYPE html>`. Cette mention aide le navigateur à savoir bien exploiter le contenu du fichier. La balise correspondante est unique, c'est-à-dire qu'il n'y a pas de balise fermante `</doctype>`. Pour l'instant, tu peux totalement ignorer ce genre de balises.

Se donner le droit d'éditer le fichier !

Tu en sais maintenant assez pour personnaliser la page d'accueil. Mais sommes-nous vraiment prêts ? Non.

Le fichier HTML qui contient la page d'accueil ne t'appartient pas : tu n'as donc pas le droit de le modifier, car il appartient au super-utilisateur `root`.

Pour pouvoir sauvegarder tes modifications, il faut intervenir sur les droits d'accès. Une première solution consiste à utiliser le Gestionnaire de fichiers :

1. Tu dois te trouver dans le sous-répertoire qui contient le fichier HTML. Ouvre le menu `Outils` du Gestionnaire et choisis la commande `Ouvrir le dossier actuel dans un terminal` (Figure 14.6).

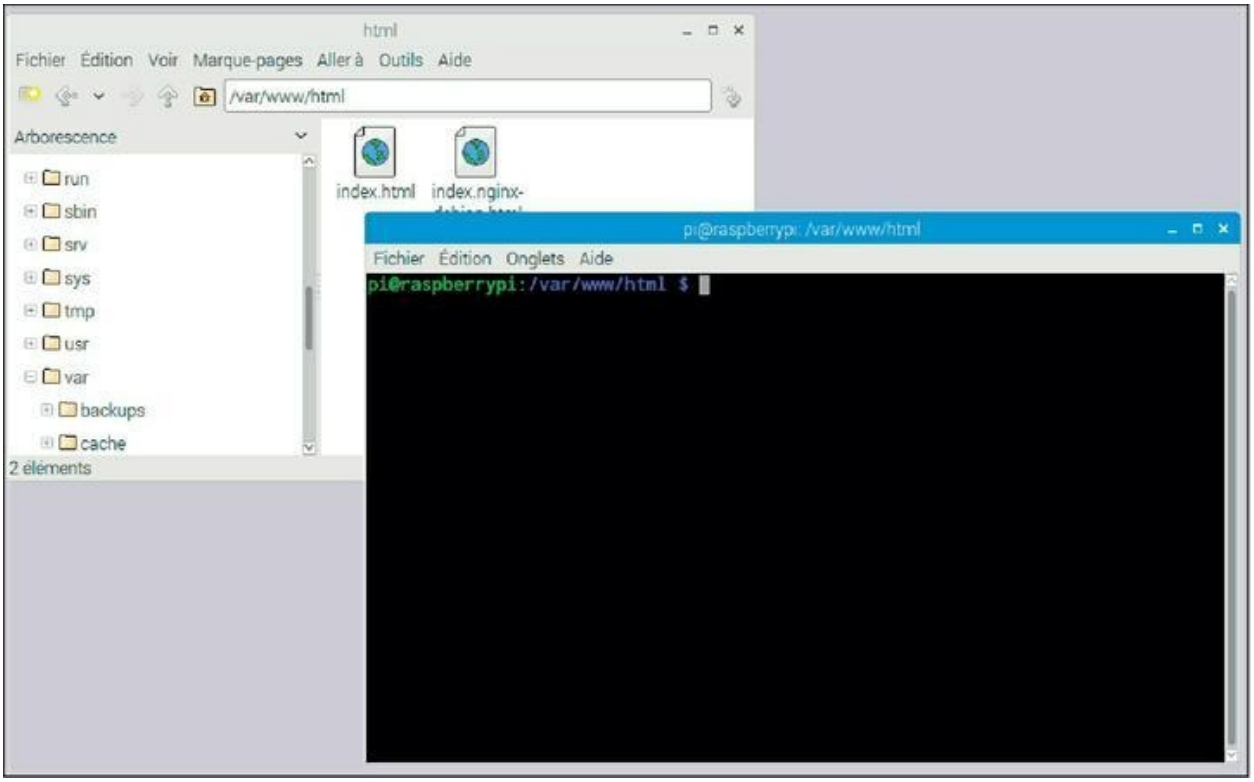


Figure 14.6 : Accès direct à un sous-répertoire avec le Gestionnaire de fichiers.

Cette technique évite de devoir descendre dans le sous-répertoire par des commandes successives `cd` dans le terminal.

Nous pouvons maintenant utiliser la commande `chown` de changement de propriétaire avec le préfixe `sudo` pour nous octroyer des droits d'accès à ce fichier.

2. Saisis donc la commande suivante et valide par Entrée :

```
sudo chmod a+w index.html
```

Dorénavant, le fichier mentionné peut être modifié par toi. La commande `Save` (Enregistrer) de l'éditeur va donc dorénavant fonctionner.

Personnaliser la page

Est-ce que tu sais où tu peux intervenir dans le fichier ? Voici la balise qui correspond au titre principal de la page (pas le titre de fenêtre marqué par `<title>`) :

```
<h1>Welcome to nginx on Debian!</h1>
```

Il suffit de modifier le texte qui se trouve entre les balises. Voici par exemple comment tu peux le modifier :

```
<h1>Bienvenue dans nginx !</h1>
```

Fais bien attention à ne pas supprimer un chevron ouvrant. Dans l'éditeur, ouvre le menu `File` et choisis la commande `Save`. Bascule dans la fenêtre du navigateur et demande une réactualisation de la page (touche F5). Le serveur va te renvoyer le nouveau contenu de la page, comme le montre la Figure 14.7.

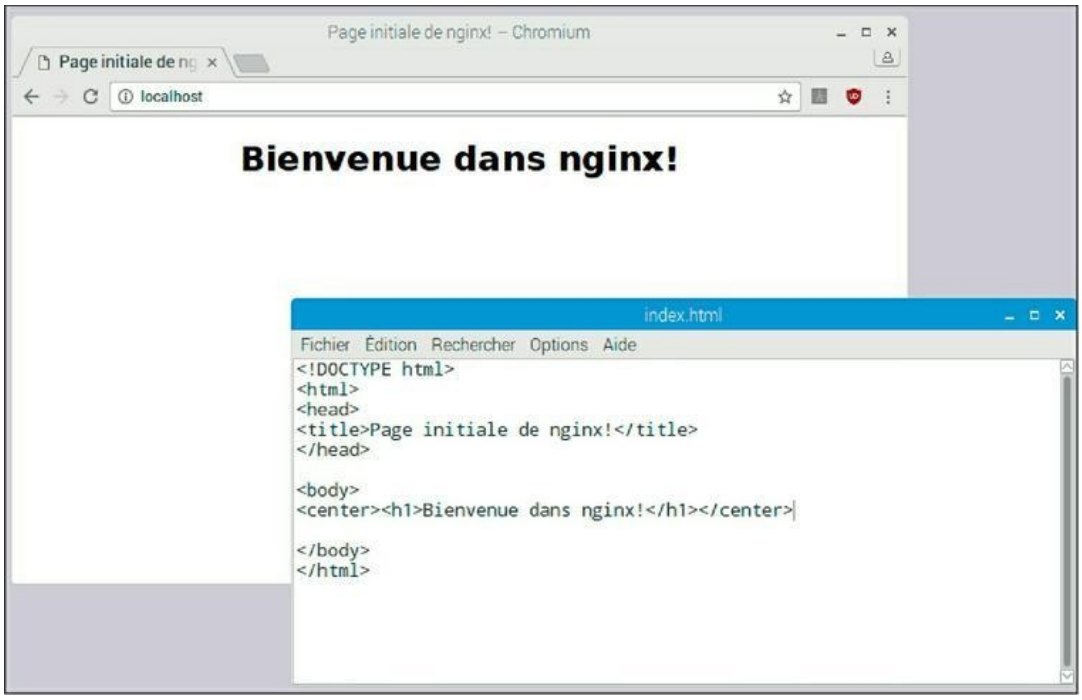


Figure 14.7 : La page d'accueil modifiée.

Le contenu et l'apparence d'une page Web

Voyons comment rendre un peu plus attrayante la page que nous venons de personnaliser.

Il faut d'abord comprendre à quoi servent les autres balises :

- La section <body> correspond au corps de contenu.
- Le couple de balises <h1> et </h1> demande de considérer son contenu comme un titre, en anglais heading, d'où le h.
- Pour réduire la taille du type, il suffit de remplacer ces balises par des balises <h2>. Tu peux descendre ainsi jusqu'au niveau cinq ; les balises <h5> donnent

> un titre avec des caractères de la même taille que le texte normal.

Pour des pages Web simples, il n'y a pratiquement aucune autre balise à maîtriser.

Séparer le contenu du décor

Une règle fondamentale de la création Web est qu'il faut toujours conserver séparés les contenus et les instructions qui vont en contrôler l'apparence, dans deux fichiers distincts.

Pour l'instant, notre fichier `.html` mélange le contenu, les définitions des sections et les éléments d'aspect. Une page Web bien construite va réunir les textes, les liens vers des images et vers d'autres sites et d'autres contenus, c'est-à-dire ce que le visiteur va voir.

Les instructions qui contrôlent la couleur, le format, les cadres, le choix des polices et certaines animations doivent être réunies dans un fichier complémentaire. Ce fichier de style porte l'extension `.css`. C'est lui que le navigateur va utiliser pour décider comment afficher le contenu.

Le soin apporté à l'aspect visuel des pages Web revêt une énorme importance. Une page qui présente bien attirera plus de visiteurs. Par ailleurs, cette séparation entre styles et contenus facilitera la mise à jour du site Web.

Pour une démonstration magistrale de l'intérêt qu'il y a à bien séparer contenu et contenant, rends-toi sur le site `csszengarden.com`. Le site est en anglais, mais cela n'a ici aucune importance. Il faut savoir que toutes les pages ont exactement le même contenu, c'est-à-dire qu'elles utilisent toutes le même fichier `.html`. Les différences de style ne sont définies que dans des fichiers CSS, qui utilisent à leur tour des illustrations. Tu constates que les règles de style offrent des possibilités incroyables.

CSS signifie *Cascading Style Sheets*, c'est-à-dire que ce sont des feuilles de styles imbriquées en cascade les unes dans les autres. Sans entrer dans les détails, il suffit de savoir que cela permet par exemple de choisir une police pour tout le contenu de <body>, et de la mettre en gras seulement pour une balise qui se trouve dans <body>, le contenu de cette sous-balise héritant du nom de la police définie au niveau supérieur et y ajoutant du gras.

Adopter la technique CSS

Nous allons créer un fichier CSS très simple afin de contrôler l'aspect de la page d'accueil. Quatre étapes sont à prévoir :

1. Création du fichier vide et modification des droits d'accès.
2. Définition de quelques règles de style dans ce fichier et enregistrement.
3. Modification du fichier HTML de façon à supprimer toutes les balises qui concernent les styles.
4. Ajout d'une directive dans le fichier HTML pour qu'il aille lire les instructions de style dans le fichier CSS.

Voyons ces quatre étapes l'une après l'autre.

Créer le fichier CSS vide

Nous allons commencer par créer un fichier vide dans la fenêtre de terminal. Tu dois toujours te trouver dans le sous-répertoire qui contient déjà le fichier `index.html`. Saisis tour à tour les deux commandes suivantes :

```
sudo  touch  mon.css
sudo  chmod  a+w  mon.css
```

Tu dois voir apparaître le fichier `mon.css` dans le répertoire. Puisque c'est toi qui l'as créé, tu disposes de tous les droits pour le modifier.

Si tu avais refermé la fenêtre de terminal, il faut à nouveau naviguer jusqu'au sous-répertoire depuis le Gestionnaire de fichiers puis choisir la commande `Outils` et ouvrir le dossier actuel dans un terminal.

Le langage CSS des règles de style

Tu peux définir des règles dans le fichier CSS pour chacune des balises présentes dans le fichier HTML. Tu peux donc associer un style à la balise `<body>` ou à la balise `<h1>`.

Évidemment, CSS est un autre langage, qui n'a aucun rapport avec le langage HTML. Le concept principal est que les règles portent des noms, et que les noms de règles sont toujours suivis d'une accolade ouvrante, de l'ensemble des directives puis d'une accolade fermante. Voici la structure générale d'une pseudo-règle (ce n'est pas une vraie règle) :

```
nom_style {
  instruction de style;
  autre instruction de style;
  encore une instruction de style;
}
```

Insérer des règles CSS

Pour savoir comment écrire des règles de style, il faut se procurer un guide de référence CSS, où en trouver un sur Internet. Le choix est vaste, même en français.

Le nombre d'options est énorme et personne ne peut tout mémoriser. Je te conseille vraiment de te choisir une bonne référence en ligne dans laquelle tu pourras puiser.

Pour commencer en douceur, définissons des règles basiques pour changer la couleur de fond de fenêtre et le corps des caractères du titre.

Voici les deux règles CSS dans le fichier :

Listing 14.2 : Contenu du fichier mon.css

```
body {
  background-color: rgb(255, 255, 0);
}

h1 {
  text-align: center; font-size: 100px;
}
```

En français, cela donnerait ceci :

- Régler la couleur de fond de page au jaune.
- Centrer le contenu des balises <h1> et utiliser un corps de 200 pixels de haut.

Après avoir saisi les deux règles, enregistre le fichier et ferme-le. Nous allons passer au fichier HTML.

Enlever les styles dans le fichier HTML

Puisque nous avons défini des règles CSS, nous n'avons plus besoin des options de style dans le fichier de base. Charge donc le fichier

`index.html` dans l'éditeur.

Dans la section du corps `<body>`, supprime tout ce qui est inutile et laisse le fichier ouvert. La section du contenu doit se résumer à ceci une fois le nettoyage terminé :

```
<body>
<h1>Bienvenue dans nginx !</h1>
</body>
```

Ne supprime pas une balise par mégarde.

Ajouter une directive de lecture du fichier CSS

Il ne reste plus qu'à ajouter une directive dans le fichier HTML pour qu'il aille lire les règles dans le fichier CSS. Place-toi au début du fichier HTML, et insère une ligne juste sous la balise ouvrante de la section de tête `<head>`. Insère alors cette directive en faisant attention à la saisie :

Listing 14.3 : Directive d'inclusion dans index.html.

```
<link rel = stylesheet type=text/css
href=my.css>
```

Cette directive indique au navigateur qu'il doit charger le contenu du fichier CSS mentionné et appliquer les règles qu'il contient aux balises du fichier HTML. Lorsque le fichier est indiqué de cette manière, il va être cherché dans le même répertoire que le fichier HTML.

Tester la nouvelle version

Pour juger du résultat, il suffit de réactualiser la page dans le navigateur. Tu devrais voir quelque chose dans le style de la Figure 14.8. Tu vois que le fond est devenu jaune et que le texte du titre est très gros. Dans la capture, j'ai laissé en surimpression les deux fenêtres du code source HTML du code source CSS.

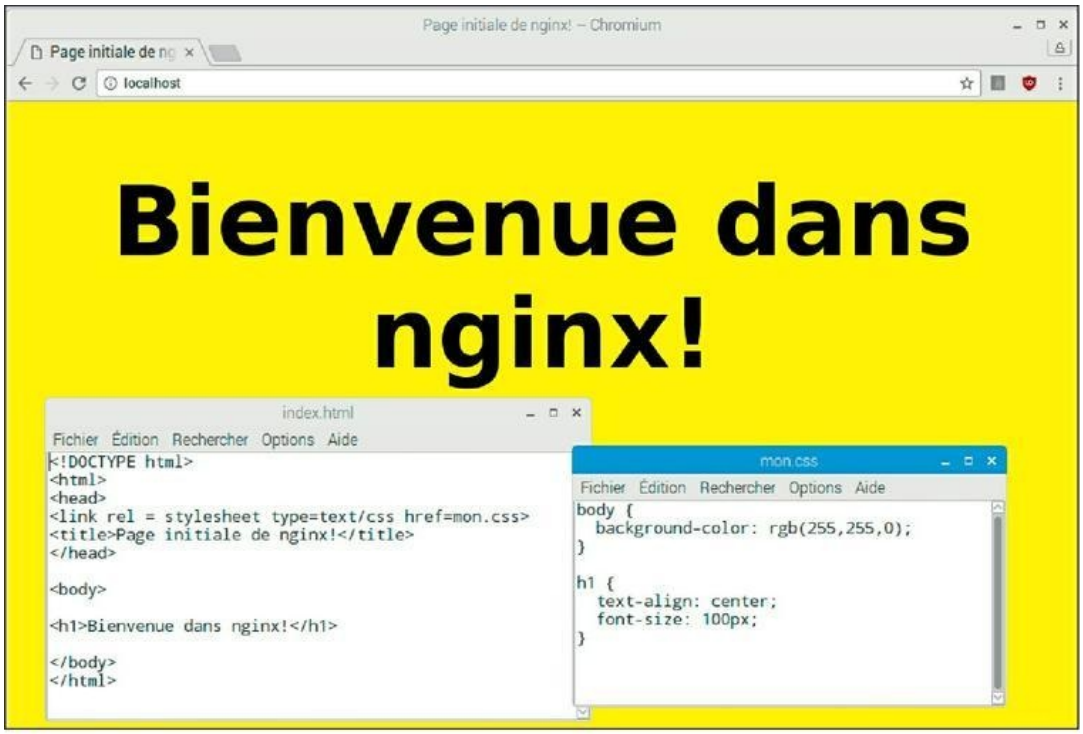


Figure 14.8 : Deuxième version de la page Web.

Tu en sais maintenant assez pour faire d'autres essais. Modifie par exemple les valeurs des trois composantes de couleur pour la couleur de fond de fenêtre. Tu peux même essayer de passer la taille du texte à 1000 px ou à

2 px. Sais-tu ce que sera le résultat ?

N'hésite pas également à modifier les règles dans le fichier CSS. Tu peux en ajouter de nouvelles en t'inspirant des exemples trouvés sur Internet.

Plus loin en CSS et en HTML

Fais une pause avant de terminer ce chapitre si les pages précédentes t'ont paru un peu denses.

Tu trouveras en librairie des livres entiers sur le CSS et le HTML, mais quelques simples conseils peuvent t'aider à démarrer. Tu peux améliorer grandement l'intérêt de tes premières pages Web, notamment en apprenant comment insérer un lien hypertexte, une illustration et des caractères spéciaux.

Quatre balises indispensables

Le Tableau 14.2 propose quelques balises incontournables. Par exemple, pour que ta page contienne un lien vers un autre site, il faut insérer la balise suivante avec son contenu :

```
<a href = http://www.qwant.com>Lien vers un
nouveau moteur de recherche.</
a>
```

Tableau 14.2 : Quelques balises HTML.

Balise	*Utilisation*
	Pour délimiter un paragraphe de texte qui sera suivi d'espace avant le texte suivant.
texte	Pour créer un lien Web. Tu remplaces la mention URL par l'adresse complète. Seule la partie « texte » sera affichée.
<img src = URL></img>	Pour insérer une illustration ou une photographie à partir du chemin d'accès URL.

 	Pour forcer le passage du texte à la ligne suivante. Tu n'as pas besoin de prévoir une balise fermante.

Le tableau suivant présente la manière de coder certains caractères spéciaux. Tu ne peux en effet pas les saisir directement parce qu'ils ne seront pas affichés. Il faut utiliser ces codes spéciaux qui commencent tous par le signe & et se terminent par un point-virgule.

Tableau 14.3 : Quelques codes de caractères spéciaux HTML.

Code	*Résultat*
	Espace insécable
©	Symbole de copyright
°	Symbole du degré de température
<	Chevron gauche
&rt;	Chevron droit

Le code de l'espace insécable (nbsp = *Non-Breaking SPace*) est indispensable parce que la norme HTML est de ne pas afficher les espaces inutiles. Si tu insères par exemple 10 espaces entre deux mots, il ne va en rester qu'une seule à l'affichage. Pour que les espaces soient conservées, il faut répéter le code   ; .

La balise et les classes auteur

Le langage HTML prédéfinit un certain nombre de balises, mais cela ne conviendra pas à tous les besoins.

La bonne nouvelle, c'est que le HTML est extensible : tu peux définir de nouvelles balises en utilisant la balise de division `<div>` puis en définissant une classe. Il s'agit du nom pour regrouper un certain nombre d'objets auxquels seront appliquées les mêmes règles CSS.

Tu dois d'abord définir la classe `<div>` dans le fichier HTML, un peu comme ceci :

```
<div class=mon_texte>Absolument fabuleux !
</div>
```

Ouvre ensuite le fichier CSS et ajoute à la fin le bloc de définition de classe suivante :

Listing 14.4 : Définition d'une classe de styles.

```
.mon_texte {
  text-align: center;
  font-size:  25px;
}
```

Note bien la présence du point initial au début du nom de la classe `.mon_ texte`. C'est grâce à ce point que le navigateur sait qu'il s'agit d'une règle qui s'applique à une classe.

Pour les noms des classes et des styles, tu as toute liberté, à condition de ne pas utiliser de lettres accentuées ni d'espace. Il faut que le nom de la règle de classe commence par un point suivi d'une lettre.

La Figure 14.9 montre comment on peut modifier notre page d'exemple en ajoutant une classe `<div>` pour le corps de texte. J'en ai profité pour changer la couleur du fond de fenêtre.

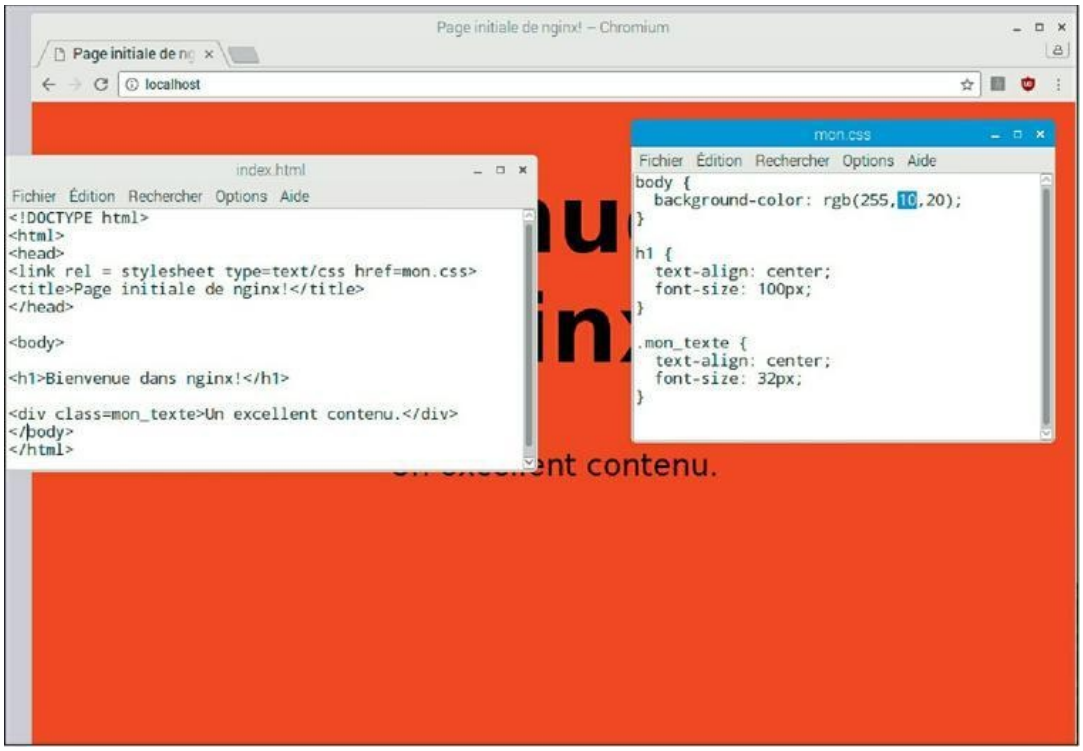
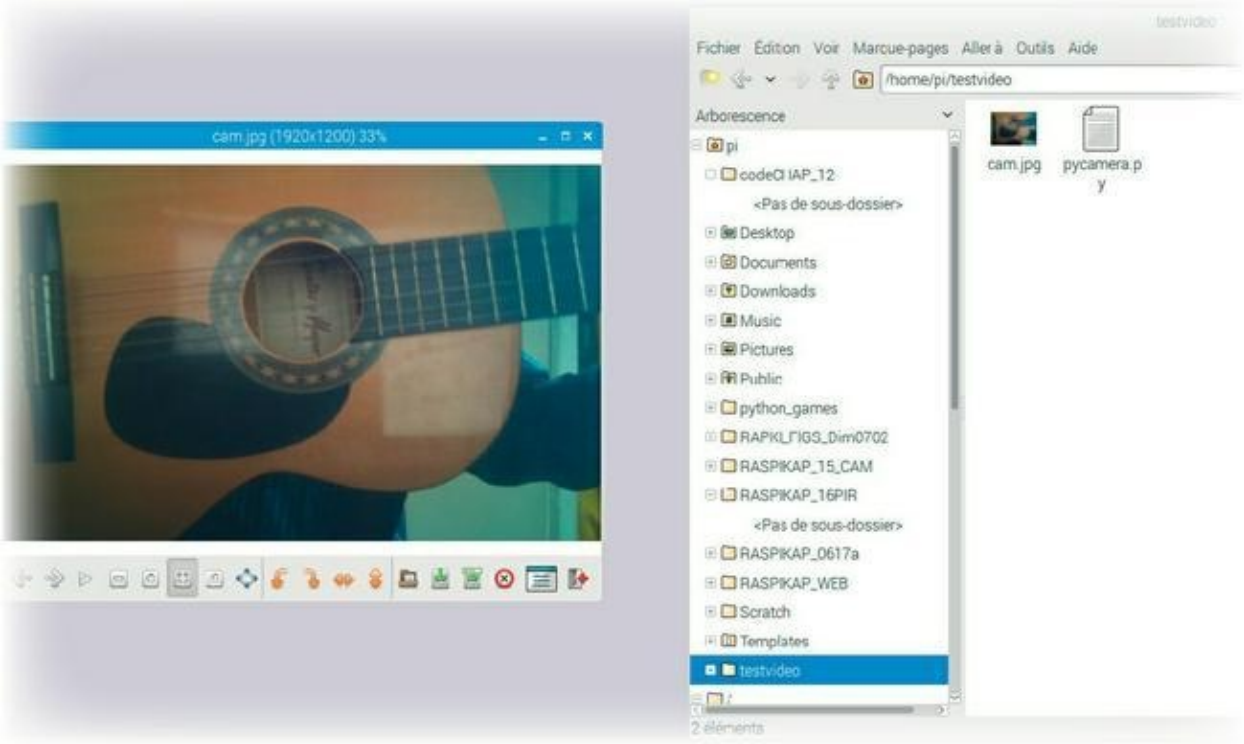


Figure 14.9 : Effet de la définition d'une classe `<div>`.

Pour aller encore plus loin dans le contrôle de l'aspect de tes pages, tu peux chercher du côté du modèle de boîte CSS, *box model*. Il permet de positionner où tu veux dans la page des blocs de texte et des illustrations. Cela dit, l'utilisation de cette technique n'est pas des plus simples, mais c'est une piste pour poursuivre la découverte de la création de pages Web.

Regarder le monde



Au programme de cette partie :

Chapitre 15
Prendre une photo avec le Raspi

N'as-tu pas envie de te servir de ton Raspberry Pi pour prendre des photos ? C'est très facile, et tu peux réutiliser une webcam dont tu n'as plus besoin, ou bien adopter la caméra qui a été conçue exprès pour le Raspi.

Une webcam

Le principe même d'une webcam et de produire un flux de données vidéo. Mais ce flux n'est pas très simple à gérer, et va vite saturer

l’espace de stockage de ta carte mémoire si tu n’y prends garde.

Nous allons dans ce chapitre nous contenter de prendre une photographie de ce que voit la caméra. Pour contrôler le périphérique, nous allons écrire un programme en langage Python. Si tu choisis une webcam sur port USB, il n’y aura rien d’autre à prévoir. Elle sera reconnue d’office, si tout va bien.

En revanche, si tu choisis de t’équiper de la Raspberry Pi Camera, il y aura un module à ajouter au système, une sorte de pilote. Cela supposera une ligne de commande supplémentaire à exécuter après chaque redémarrage quand tu voudras utiliser la caméra.

Choisir une webcam

De nos jours, quasiment tout le monde dispose d’une webcam, car c’est ce qui permet de communiquer en vidéo avec des logiciels tels que Skype ou FaceTime.

Tu trouveras facilement une webcam dans les 20 à 30 €. Le modèle que j’ai essayé est la Logitech C270HD.

Installer la webcam

Soit la webcam est reconnue, ce qui sera presque toujours le cas, soit elle ne l’est pas. Il suffit de la brancher sur un port USB libre et de sauter la section suivante consacrée à la caméra Raspberry Pi.

La caméra Raspberry Pi

Si tu choisis la caméra Raspberry, il faut d’abord la mettre en place physiquement en connectant son câble à la carte Raspi puis activer une option système.

1. Tout d’abord, assure-toi d’avoir déchargé toute ton électricité statique en touchant un radiateur ou une conduite d’eau.

2. Tu as bien sûr mis ton Raspi hors tension. N'hésite pas à débrancher toutes les connexions pour être plus à l'aise.

3. Le Raspi comporte deux connecteurs avec un étrier blanc. Observe celui qui se trouve juste derrière les prises Ethernet et USB, entre la prise HDMI et la prise Jack (Figure 15.1).

Figure 15.1 : Le connecteur pour brancher la caméra Raspberry.

4. Tire un peu cet étrier blanc vers le haut avec un ongle dans chaque angle pour pouvoir le basculer ensuite un peu en arrière. Le but est d'ouvrir l'espace dans lequel tu vas insérer le câble en nappe de la caméra.

5. Déballe la caméra et observe la différence d'aspect des deux côtés de l'extrémité de son câble.

6. Prends le câble en nappe entre deux doigts en orientant les contacts du côté de la prise HDMI, donc en opposition aux prises USB. Glisse doucement le connecteur dans les contacts de la prise jusqu'à toucher le fond (Figure 15.2). Les contacts métalliques ne doivent dépasser que d'un bon millimètre.

Figure 15.2 : Insertion du câble.

7. Maintiens le connecteur bien au fond de la prise en continuant à pousser légèrement et repousse l'étrier blanc contre les contacts puis repousse-le vers le bas, ce qui va verrouiller le câble en place (Figure 15.3).

N’hésite pas à appuyer sur l’étrier. En tirant doucement sur le câble, il doit rester en place.

Figure 15.3 : Verrouillage du câble.

8. Vérifie que le câble est bien inséré perpendiculairement à la carte.

   Repositionne le petit circuit de la caméra au bout du câble de telle sorte qu’il n’y ait aucun contact avec une pièce métallique, ni bien sûr avec aucun composant du Raspi.

   Ta caméra est connectée.

Configuration du module Caméra

Du fait que la caméra Raspberry Pi n’est pas branchée sur un port USB, elle ne sera détectée par le système qu’une fois le module

correspondant chargé en mémoire.

Il y a deux étapes à prévoir :

1. Activation de l'option vidéo.
2. Chargement dans le noyau du module approprié.

Pour activer l'entrée vidéo :

1. Tu peux redémarrer ton Raspi si ce n'est pas encore fait.
2. Ouvre le menu général, choisis le sous-menu `Préférences` puis `Raspberry Pi Configuration` (Figure 15.4).
3. Accède alors à la deuxième page, `Interfaces`, et active le bouton radio `Caméra` puis referme la boîte de dialogue par OK.

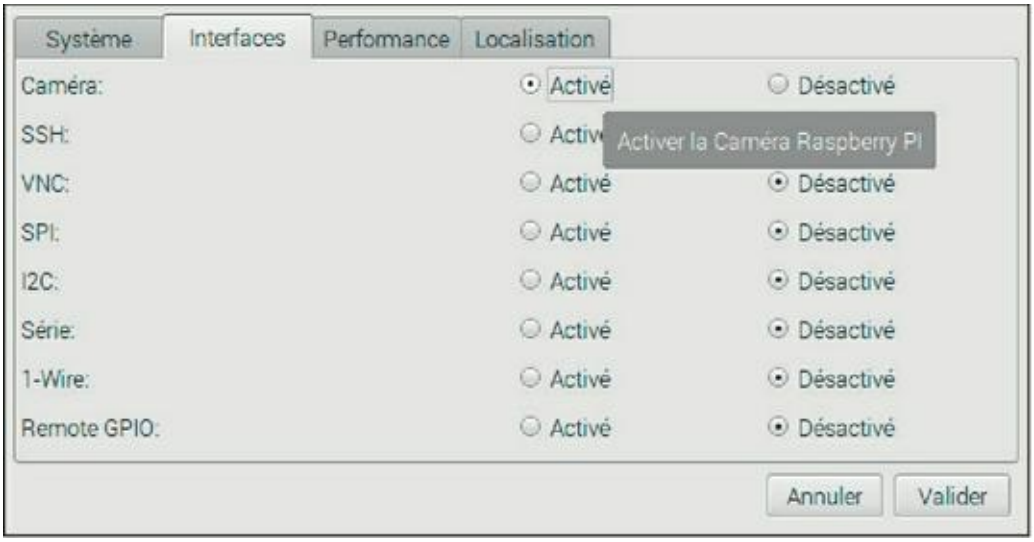


Figure 15.4 : Activation de la fonction Caméra.

Dorénavant, la caméra sera toujours autorisée. Ce réglage est mémorisé pour les prochains

redémarrages.

Pour charger le module système :

1. Ouvre une fenêtre de terminal (LXTerminal) et saisis la commande suivante :

```
sudo modprobe bcm2835-v412
```

2. Vérifie bien ta saisie avant de valider par la touche Entrée. Tu devras répéter cette commande après chaque redémarrage du Raspi pour que la caméra soit reconnue.

La commande `lsmod` permet de voir la liste des modules chargés.

Ce chargement de module est bien sûr inutile si tu utilises une webcam USB.

Si tu comptes utiliser souvent la caméra, tu peux ajouter la commande que nous venons de voir dans un fichier qui est exécuté à chaque démarrage. Ce fichier porte le nom `rc.local` et se trouve dans le sous-répertoire `/ etc`. Je rappelle que pour modifier ce fichier, tu dois disposer des droits du super-utilisateur root. Tu peux utiliser l'éditeur en mode texte nano en le lançant de la manière suivante depuis une fenêtre de terminal une fois dans le sous-répertoire `/etc` :

```
sudo nano rc.local
```

Les raccourcis clavier des commandes d'édition sont rappelés en bas de fenêtre.

La librairie Pygame

Si tu devais toi-même écrire le programme pour récupérer les données binaires depuis la caméra et en produire un fichier, tu n'es pas au bout de tes peines. De plus, cela ne marcherait qu'avec un ou deux modèles de périphériques, car ils sont tous différents les uns des autres.

Il y a heureusement une solution simple. En effet, le langage Python possède une librairie complémentaire qui porte le nom Pygame (`http://pygame.org`). Elle est installée d'office dans Raspbian.

Le domaine d'emploi principal de la librairie Pygame est la création de jeux vidéo. Elle comporte par exemple des fonctions pour dessiner des blocs dans une fenêtre, pour contrôler les mouvements des objets graphiques et pour détecter les collisions entre deux objets à l'écran. C'est une sorte de version plus puissante de Scratch, pour Python.

Ce qui nous intéresse dans la librairie Pygame est une série de fonctions pour gérer une caméra afin de prendre des photos et même des vidéos. Cette famille de fonctions est utilisable avec la plupart des webcams, et bien sûr avec la caméra officielle du Raspberry Pi. Il suffit de quelques instructions pour faire fonctionner le projet.

Cela dit, le module de capture n'est pas très puissant. Avec la plupart des webcams, tu ne pourras que prendre une photo, pas une vidéo. L'avantage est qu'il n'y a aucun autre pilote ou complément à prévoir.

La librairie Pygame est une excellente excuse pour approfondir tes connaissances en langage Python. Tu ne peux pas l'envisager pour créer un jeu vidéo commercialisable, mais cela suffit largement pour acquérir les grands principes de création d'un jeu vidéo en Python. De plus, le site Web de Pygame propose des centaines d'exemples.

Ajouter Pygame au projet

Pygame fait partie des librairies standard de Python, c'est-à-dire qu'il n'y a pas besoin de l'installer. En revanche, si ton programme a besoin d'utiliser une fonction de cette librairie, il faut le dire au début du code source de ton programme. La technique la plus simple consiste à utiliser la commande `import`, suivie du nom du module :

```
import pygame
```

Dans notre exemple, nous avons également besoin d'importer un sous-ensemble spécifique qui est celui pour la caméra. Il faut donc également ajouter la ligne suivante :

```
import pygame.camera
```

Comment peut-on deviner quelle sous-fonction il faut importer ? Il n'y a pas de logique particulière. Il faut s'inspirer d'exemples écrits par d'autres et lire la documentation.

Initialiser la caméra

En général, avant de pouvoir utiliser un module, il faut l'activer, ce qui est habituellement réalisé par un appel à une fonction qui porte le nom `init()`, ce qui signifie initialisation.

Un module est un réservoir de fonctions. Il peut en contenir des dizaines ou des centaines. Chacune d'elles effectue un traitement spécifique.

Pour utiliser une fonction, il faut d'abord indiquer le nom du module, puis un point puis le nom de la fonction. Parfois, il faut indiquer un troisième nom derrière les deux premiers.

Voici les deux lignes qui permettent d'activer la caméra :

```
pygame.init()
pygame.camera.init()
```

Il n'y a pas de majuscule au nom `pygame` et il n'y a pas d'accent au nom `camera`.

Choisir le format d'image

Avec Pygame, tu peux choisir les dimensions de l'image que tu vas prendre. Bien sûr, tu es limité par la résolution maximale de la caméra, mais pour tes premiers essais, autant choisir des dimensions inférieures, car la prise de vue sera plus rapide. De plus, le fichier résultant occupera moins d'espace.

Pour définir la largeur et la hauteur de l'image, nous allons profiter de la possibilité de créer deux variables que nous allons appeler `largeur` et `hauteur`.

Voici donc les deux lignes suivantes du code source que nous sommes en train d'écrire progressivement :

```
largeur  = 640
hauteur = 480
```

Créer une instance de caméra

Nous allons maintenant découvrir un concept très important en programmation : la programmation orientée objets. En effet, dans la librairie que nous avons importée, il y a la définition d'une classe de caméra. Pour en profiter, il nous faut demander la création d'un objet à partir de cette classe, c'est-à-dire d'une instance.

Un objet est la réunion de plusieurs fonctions et de plusieurs variables dans une coquille qui les protège.

Pour utiliser l'objet, nous citons son nom puis un signe point puis le nom de la fonction dont nous avons besoin. Lorsqu'une fonction appartient à un objet, on l'appelle une *méthode*.

Pour travailler, l'objet caméra de Pygame doit réaliser un certain nombre d'opérations de préparation assez complexes. Comme c'est un objet, nous n'avons pas besoin de nous en soucier. Nous allons nous contenter des valeurs qui sont réglées par défaut au départ.

Les seuls paramètres que nous devons choisir son la largeur et la hauteur et le chemin d'accès au périphérique vidéo qui correspond à la caméra.

Qu’est-ce que c’est que ce chemin d’accès à une vidéo ? Sous Linux, tous les périphériques sont réunis dans un répertoire spécial qui porte le nom `/ dev`. Si tu accèdes avec une fenêtre de terminal à ce sous-répertoire et que tu demandes l’affichage de son contenu avec la commande `ls -l`, tu verras quelque chose dans le style de la Figure 15.5.

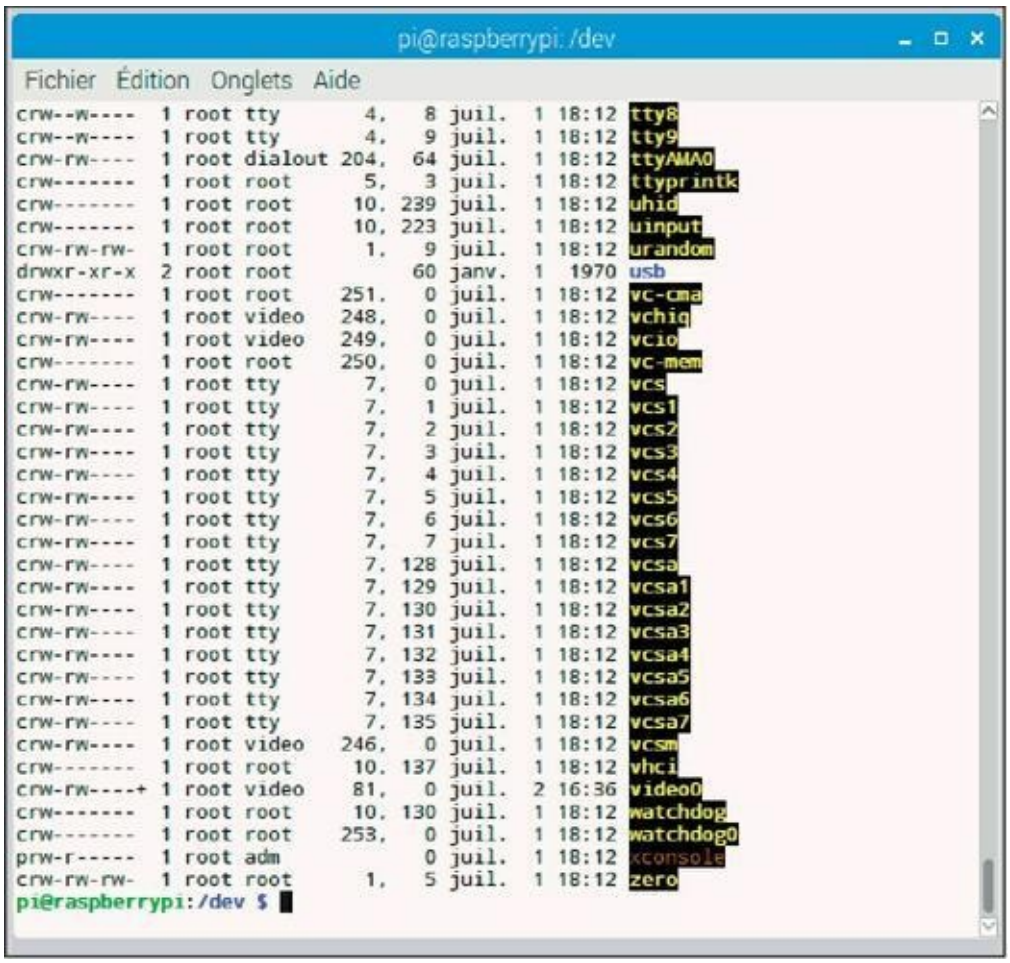


Figure 15.5 : Affichage du contenu du répertoire /dev.

Si tu as utilisé la commande `sudo modprobe` comme demandé plus haut, tu dois voir parmi les derniers noms affichés le fichier `video0`. Nous pouvons donc utiliser comme source des données vidéo le chemin d’accès `/dev/ video0`.

Voici à quoi ressemble l'instruction qui permet de créer notre objet de la classe `Camera` qui porte le nom `cam` :

```
cam = pygame.camera.Camera(«/dev/video0»,
(largeur, hauteur))
```

J'ai déjà dit dans un chapitre précédent que sous Linux, tout était fichier. Le flux vidéo qui provient d'un périphérique est donc aussi un fichier. Particularité : c'est simplement un fichier dont la taille est inconnue et dont tu peux récupérer le contenu en branchant une fonction de lecture sur le tuyau. Les droits d'accès aux fichiers de `/dev` sont les mêmes que ceux des fichiers normaux.

Si tu regardes la première colonne dans l'affichage du contenu de `/dev`, tu constates qu'il y a une lettre `c` en première position. Cela signifie qu'il ne s'agit ni d'un fichier, ni d'un répertoire (d), mais d'un périphérique de type caractère (c). Tu constates également le T majuscule qui est un droit d'accès spécial pour les éléments qu'il ne faut pas pouvoir supprimer.

Les instructions de prise de vue

Notre caméra est maintenant prête. Il ne reste plus qu'à ajouter les instructions pour prendre la photo. Elles sont au nombre de trois :

```
cam.start()
image = cam.get_image()
cam.stop()
```

La première instruction demande au périphérique de se tenir prêt ; la suivante demande de prendre la photo pour la stocker dans une image. La dernière arrête le périphérique.

À partir de ce moment, la photo est prise. Il nous reste à enregistrer les données de l'image dans un fichier :

```
pygame.image.save(image, 'cam.jpg')
```

Tu vois comme il est simple d'utiliser cette librairie. Tu n'as pas à t'inquiéter de conversion entre le format d'origine et un des formats reconnus pour les images. Ici, nous utilisons le format JPEG.

Voici maintenant le listing complet du code source que nous venons de décrire ligne après ligne. La première ligne est un commentaire qui te rappelle qu'il faut charger le pilote pour utiliser la caméra Raspberry.

Pour disposer de ce code source, commence par créer un fichier vide dans ton répertoire personnel `/home/pi`. Tu peux utiliser la commande `touch` comme indiqué au Chapitre 11 ou démarrer directement l'éditeur de texte. Saisis alors le code source en vérifiant bien que tu n'as pas fait de faute de frappe. Enregistre le fichier sous le nom `picamera.py`.

Listing 15.1 : Code source de picamera.py

```
# sudo modprobe bcm2835-v4l2
import pygame
import pygame.camera
pygame.init()
pygame.camera.init()
largeur = 640
hauteur = 480
cam = pygame.camera.Camera(«/dev/video0»,
(largeur, hauteur))
cam.start()
image = cam.get_image()
cam.stop()
pygame.image.save(image, 'cam.jpg')
```

Je rappelle que le code source de tous les exemples de ce livre est disponible dans un fichier d'archive compressé dans la page du livre

sur le site de l'éditeur.

Tester notre code

Pour lancer l'exécution depuis l'éditeur en fenêtre graphique, utilise la touche F5. Tu peux aussi ouvrir une fenêtre de terminal, vérifier que tu es bien dans ton répertoire personnel et saisir la commande suivante :

```
python  picamera.py
```

S'il n'y a pas d'erreur, c'est que tu as tout saisi correctement. Il suffit alors de patienter quelques secondes que la photo soit prise.

Tu peux ensuite faire afficher le contenu du répertoire pour constater que le fichier image que tu as demandé de générer est bien présent (Figure 15.6).

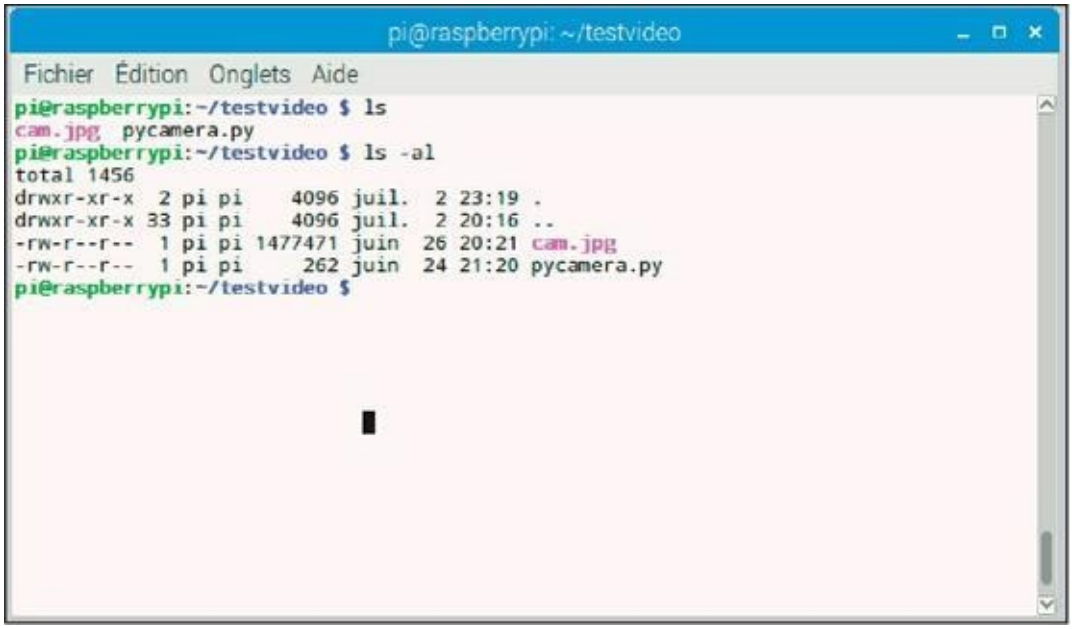


Figure 15.6 : Affichage du répertoire contenant le fichier généré.

Dans cet exemple, j'ai pris soin de créer un sous-répertoire nommé `testvideo` dans lequel j'ai placé le fichier du code source. De ce fait, le fichier image a été déposé au même endroit. Si tu travailles

directement dans la racine `/home/pi`, d'autres fichiers vont bien sûr apparaître dans la liste.

Afficher la photo

Puisque la photo a été générée dans un format reconnu, tu peux ouvrir le Gestionnaire de fichiers, aller jusqu'au répertoire contenant le fichier et double-cliquer pour faire afficher l'image (Figure 15.7).

Pour afficher l'image en grand format, double-clique dans la barre de titre.

Si tu relances le programme, la nouvelle image écrasera l'ancienne.

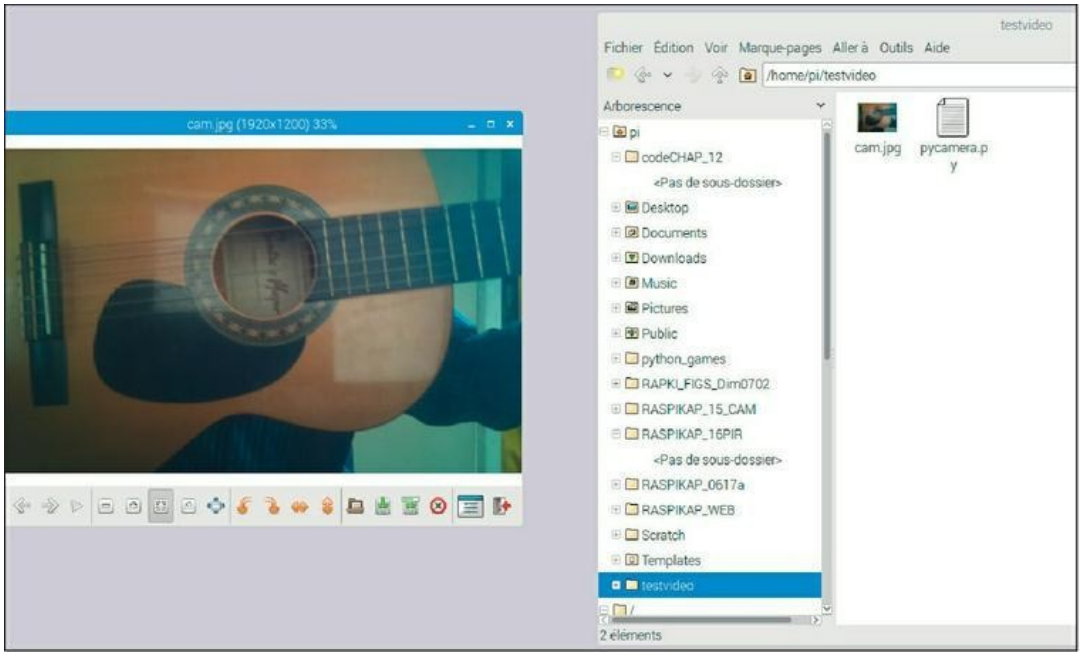


Figure 15.7 : Affichage de l'image venant d'être capturée.

En cas de problème

Si le projet ne fonctionne pas, essaye les solutions suivantes :

» Revérifie le code source. As-tu bien saisi les instructions dans le même ordre que dans le listing ?

N’as-tu pas oublié une parenthèse ?

- As-tu bien pensé à lancer la commande `modprobe` si tu utilises la caméra Raspberry ?
- As-tu bien activé le sous-système `Caméra` dans la configuration du Raspberry Pi ?
- Vérifie que tu es dans le bon sous-répertoire. Tu dois te trouver dans `/home/ pi` ou dans un de ses sous-répertoires. Si tu n’es pas dans un de ces endroits, tu risques des problèmes de droits d’accès.
- Est-ce que la caméra est bien connectée à ton circuit ? On ne sait jamais.
- Est-ce que ta caméra est en état de fonctionner ? Si tu as des doutes, essaie avec une autre.

Pour aller plus loin

Voici quelques pistes pour aller plus loin :

- Tu peux faire ajouter la date et l’heure dans le nom du fichier généré. Je montre une technique à ce sujet dans le chapitre suivant.
- Tu peux utiliser les variations de la fonction `pygame.transform()` pour faire tourner et retailler la photo avant de l’enregistrer.
- Tu peux appliquer des effets de couleur et de flou. Je te préviens cependant que l’utilisation de ces fonctions n’est pas très simple. Il faut en savoir un peu

plus au sujet de Python et du traitement des images pour y parvenir.

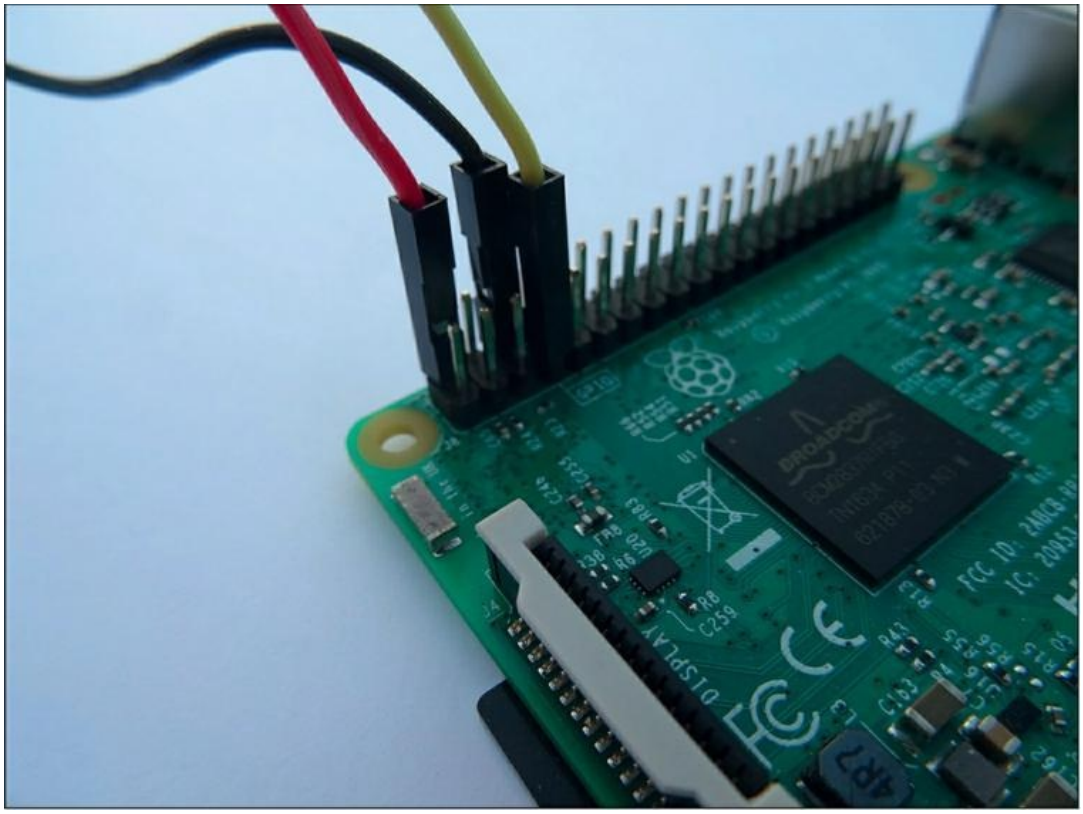
» Il serait intéressant de pouvoir décider des dimensions de l'image dans la commande de lancement du programme au lieu de les définir une fois pour toutes dans le code source. Ce sont des paramètres de ligne de commande. Tu peux chercher sur Internet l'expression « Python command-line arguments ».

Pour faire varier le comportement d'un programme en fonction des options que l'on ajoute au moment de le lancer, il faut pouvoir analyser ce qui a été saisi à la suite du nom du programme, ce qui correspond à l'opération de *parsing*. C'est la façon dont les programmes informatiques essaient de comprendre les informations qu'on leur transmet.

Chapitre 16
Qui va là ?

Au cours des chapitres précédents, tu as découvert comment utiliser ton Raspi pour toutes sortes d'activités, et notamment pour apprendre à programmer. Mais pour l'instant, ton Raspi n'est entré en interaction qu'avec toi-même : avec tes doigts, tu as utilisé le clavier et la souris ; avec tes yeux, tu as lu ou admiré ce qui est affiché à l'écran. Tu as peut-être également écouté de la musique grâce à la sortie audio.

Mais pour entrer en interaction avec le monde physique, comment faire ? C'est le sujet de ce chapitre.



Une bête de labo

Sur un ordinateur de bureau habituel sous Windows ou macOS, rien n'est prévu pour se mettre à l'écoute du monde réel et pour agir sur celui-ci. Il faut ajouter des cartes spéciales dites d'entrées/sorties, comme celles qu'utilisent les laboratoires et centres de recherche.

Bien qu'il coûte dix fois moins qu'un ordinateur classique, ton Raspi est bien plus ouvert à ce niveau. Grâce à lui, tu peux te lancer à la découverte du monde de l'expérimentation. Il dispose en effet d'un connecteur d'entrées/ sorties à usage général, appelé GPIO (*General Purpose Input-Output*). La Figure 16.1 le montre.

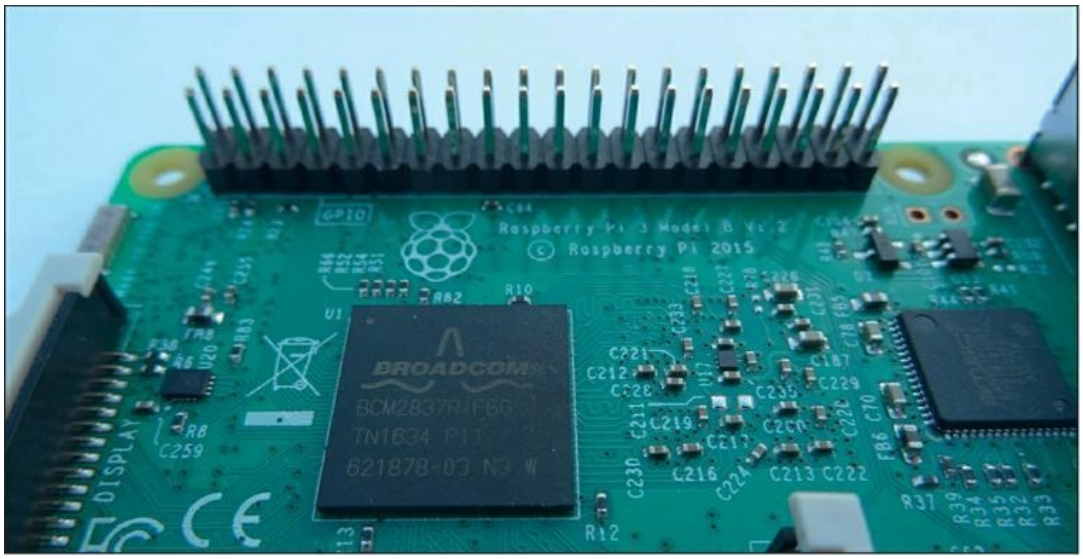


Figure 16.1 : Vue générale du connecteur GPIO.

Grâce à ce connecteur, tu vas pouvoir transformer ton Raspi en un appareil capable de réagir à son environnement physique et d'y intervenir. Tout cela devient possible en raccordant à ce connecteur des capteurs et des actionneurs.

Des organes sensoriels et des muscles

Un corps d'humain ou d'animal possède :

- des organes sensoriels pour rester informé de l'environnement dans lequel il se déplace ;
- des muscles pour se déplacer dans cet environnement.

De même, un ordinateur peut être doté d'organes sensoriels sous forme de capteurs ou senseurs. Pour agir sur son environnement, il peut être doté d'actionneurs, qui sont principalement des moteurs électriques, tels que ceux qui donnent vie aux robots.

Des capteurs pour s'informer

Le monde d'aujourd'hui est rempli de capteurs électroniques. Quelques exemples :

- le thermostat d'un radiateur électrique ;
- le système d'allumage automatique des phares d'une voiture ;
- le détecteur de mouvement d'un éclairage extérieur ;
- l'accéléromètre d'une manette de jeu vidéo ;
- la balance pour peser les bagages à l'aéroport ;
- le capteur de pression d'une usine pétrochimique ;
- le détecteur de fumée, obligatoire dans nos domiciles.

La figure suivante (Figure 16.2) montre l'aspect de plusieurs catégories de capteurs.

Cette page provient de la boutique Web de la société SemaGeek (boutique.semageek.com).

Tous ces capteurs peuvent être exploités avec ton Raspi, moyennant une phase de montage et de câblage puis une phase de programmation. Pour programmer, nous utiliserons le langage Python que tu connais maintenant.

Les capteurs permettent de simuler les cinq organes sensoriels de l'homme, et même d'en faire un peu plus. Par exemple :

- Pour l'œil, il y a des capteurs de luminosité et des analyseurs de couleur.
- Pour les oreilles, il y a bien sûr les microphones, mais également de simples capteurs de bruit ambiant.
- Pour les odeurs, il y a des détecteurs de gaz.
- Pour le toucher, il y a des détecteurs de pression.
- Pour le goût, il y a des analyseurs d'acidité.

Et tout cela sans compter les thermomètres, les hygromètres, les capteurs d'ultrasons, les détecteurs d'ultraviolets, les détecteurs de radioactivité, les récepteurs GPS, *etc*.

Voyons donc maintenant comment le Raspi peut utiliser un capteur.

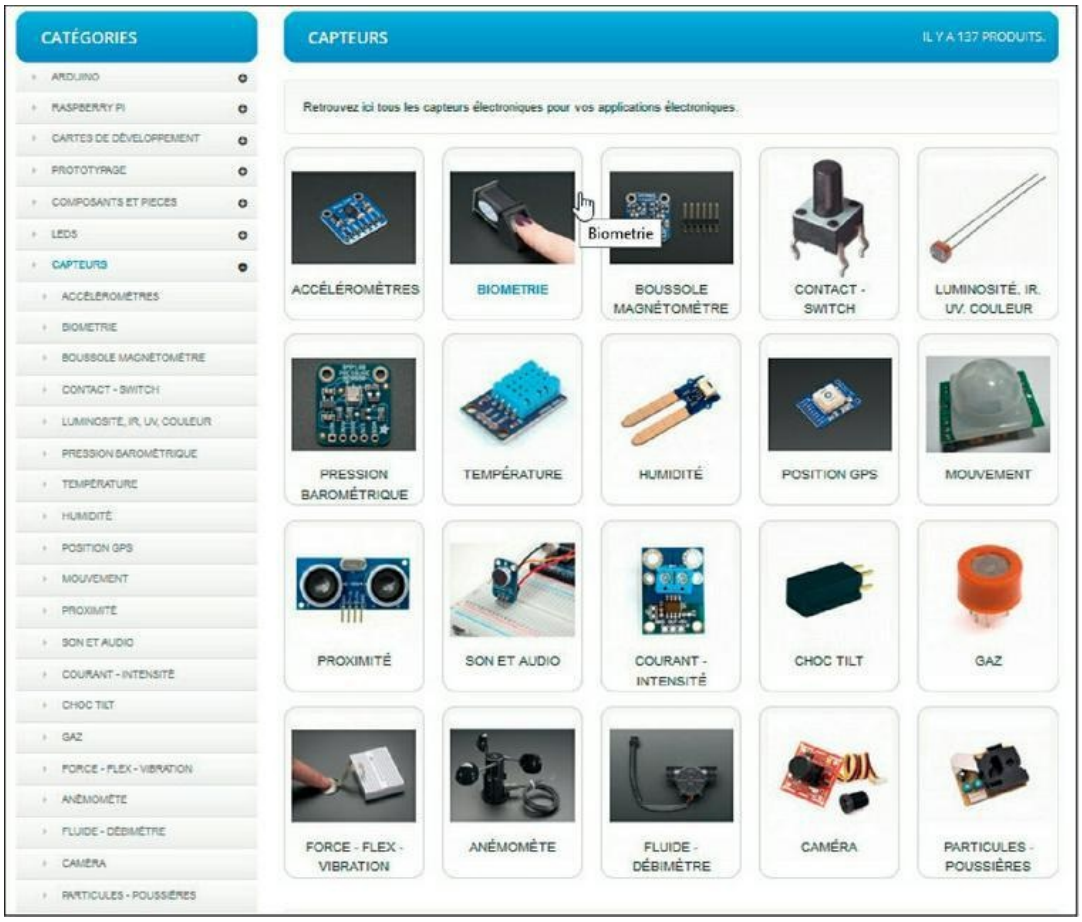


Figure 16.2 : Quelques capteurs.

Le connecteur du labo

Sur les modèles actuels du Raspberry Pi, c'est-à-dire les modèles 2 B et 3 B, le connecteur GPIO comporte 40 broches. Sur les premiers modèles, il n'y en avait que 26, mais les 26 premières des récents modèles ont exactement le même usage, ce qui les rend compatibles.

Les broches du connecteur sont de deux types (Figure 16.3) :

- alimentation en énergie ;
- données.

Figure 16.3 : Schéma du connecteur GPIO.

Alimentation

Une dizaine de broches servent à l'alimentation en énergie des capteurs.

Pour que du courant passe dans un circuit, il faut au moins deux contacts, comme sur une pile : un pôle positif et un pôle négatif.

Sur le connecteur GPIO, il y a :

- des broches avec 0 volt (il y en a 8), c'est-à-dire la masse (les points noirs dans la Figure 16.3) ;
- des broches avec du 3,3 volts positif (les deux broches 1 et 17 en orange) ;
- des broches avec du 5 volts positif (les deux broches rouges 2 et 4).

Données

Dans la Figure 16.3, les 26 broches de données sont en jaune clair. (Les deux broches blanches 27 et 28 servent à raccorder un circuit mémoire EEPROM.)

Ce sont ces broches de données qui vont servir soit à recevoir des informations depuis le monde extérieur, soit à envoyer des

informations vers l'extérieur.

Chaque broche de donnée est polyvalente : elle peut servir en entrée ou en sortie (pas les deux à la fois). C'est toi qui décides comment utiliser la broche dans ton programme.

> » Une broche utilisée en entrée va servir à détecter la présence ou l'absence d'une tension en provenance d'un capteur.
>
> » Utilisée en sortie, la broche va fournir une tension ou pas de tension à un actionneur.

Dans ce chapitre, nous n'allons utiliser qu'une seule broche de données, et nous allons l'utiliser en entrée, puisque nous allons exploiter un capteur.

C'est dans notre programme que nous allons prévoir une instruction pour régler la broche afin qu'elle serve en entrée et non en sortie.

Le Raspberry Pi est très délicat : chacune des sorties ne peut fournir au maximum que 16 milliampères (16 mA). Si tu y branches directement ne serait-ce qu'une seule diode LED, tu risques de détruire le circuit associé à la broche ou la totalité du processeur.

Un projet : le détecteur de mouvement

Pour montrer comment s'utilise le connecteur GPIO de ton Raspi, je te propose un projet à usage tout à fait pratique : un détecteur de mouvement.

J'ai choisi ce type de capteur, car il est très peu coûteux, et ne réclame aucun autre composant pour être utilisable. Il n'y a pas à rajouter de résistance ou de transistor, alors que c'est souvent le cas pour d'autres capteurs.

Le capteur PIR

Le surnom technique de ce capteur est PIR, ce qui signifie Proximité Infra-Rouge. C'est un capteur d'ondes infrarouges proches. Voici son aspect général du côté de la réception des ondes infrarouges (Figure 16.4).

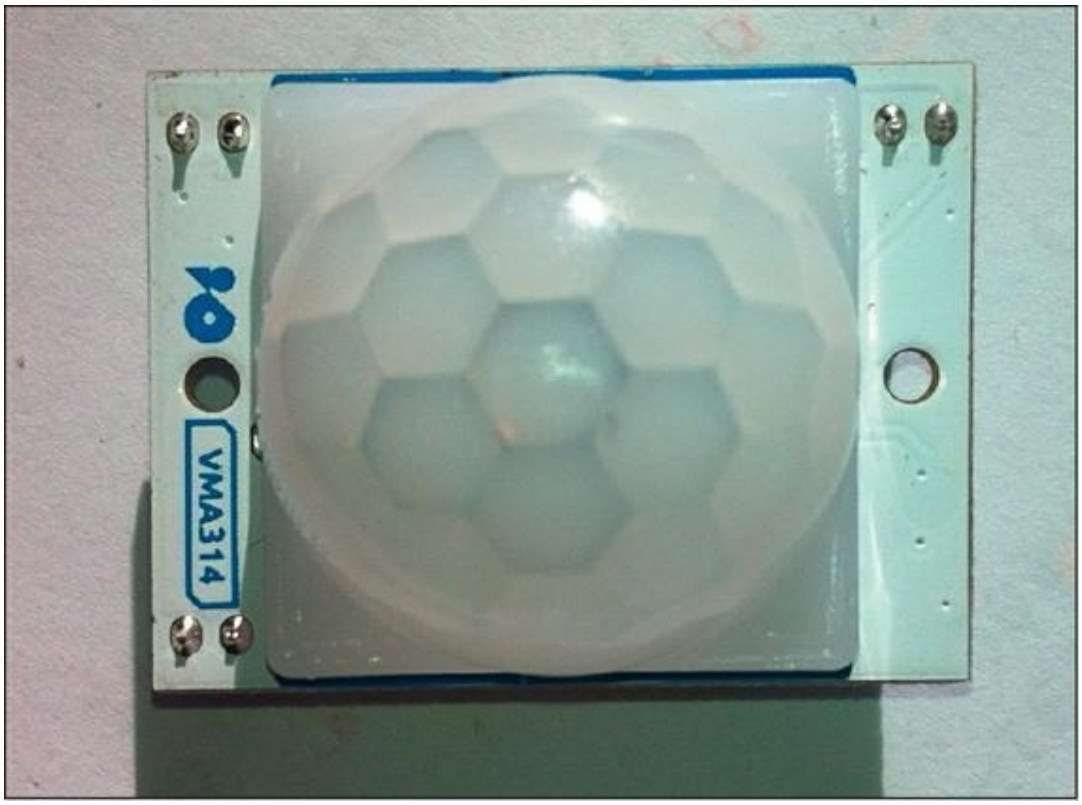


Figure 16.4 : Le détecteur de mouvement PIR.

Ce dôme à facettes en plastique opaque est en fait une lentille de Fresnel qui concentre les rayons infrarouges sur la petite puce qui est cachée au centre dessous.

Le principe de fonctionnement est le suivant :

1. Au démarrage, le capteur se construit une image de ce que voit le détecteur, ce qui lui prend quelques secondes.

2. Une fois cette préparation terminée, le détecteur est prêt à comparer l'image qu'il reçoit à chaque instant à celle qu'il a mémorisée.

3. Dès qu'il détecte une différence, c'est qu'il y a eu un mouvement dans son champ de vision, ce qui permet de changer l'état électrique de son fil de sortie qui va passer de 0 V à 3 V.

4. Après chaque détection, il faut lui laisser à nouveau quelques secondes pour reconstruire son image du paysage au calme, après quoi il est réarmé, donc prêt à la prochaine détection. La sortie est alors ramenée à 0 V.

Ce capteur PIR se branche avec trois fils dont deux servent à son alimentation en énergie. En général, on utilise un fil de couleur noire pour le 0 V ou masse, et un fil rouge pour le pôle positif, qui doit ici être du 5 V.

Le troisième fil sera le fil du signal de sortie qui va basculer entre 0 et 3 V selon que le capteur a détecté ou pas quelque chose.

Voici reproduite la fiche technique qui décrit ce capteur :

Caractéristique	*Valeur*
Tension d'alimentation :	5 V continus
Connexion :	3 broches GND, VCC et OUT
Délai de déclenchement :	Réglage par RP1
Sensibilité :	Réglage par RP2
Longueur de l'écho :	0,3 à 18 secondes

Niveau du signal de sortie :	Alarme = 3 V, Repos = 0 V
Distance de détection maximale :	7 mètres
Température de fonctionnement :	-15 à +70 °C
Angle de détection :	120°
Dimensions :	32 x 24 x 25 mm

Approvisionnement

Voici les composants et pièces à réunir pour le montage :

- Un capteur infrarouge PIR ou détecteur de mouvement (moins de 5 euros) ; choisis un modèle avec les trois broches de sortie mâles déjà soudées, par exemple le Velleman VMA314.
- Pour relier électriquement le capteur au Raspi, il nous faut des fils avec des fiches sur les extrémités. On les appelle des straps. Il nous faut trois straps femelle/femelle si possible un rouge, un noir et un autre (jaune, bleu, vert, blanc, qu'importe). Pour la longueur, choisis des straps d'au moins 10 cm de long, voire plus.
- Si tu n'as pas de straps femelle/femelle, tu peux t'en sortir avec des mâle/femelle en utilisant une plaque d'essai intermédiaire.
- Il faut ton Raspberry Pi, bien sûr.

» Aussi un tout petit tournevis plat pour régler la sensibilité du capteur.

Pour le programme, nous allons avoir besoin de fonctions prédéfinies, donc de librairies. Celle qu'il nous faut va simplifier l'utilisation du connecteur GPIO. Elle est installée d'office. Il n'y a donc aucun téléchargement et installation à prévoir (ce n'est pas toujours le cas).

Montage et câblage

Tout d'abord et avant tout, débranche ton Raspi. Il ne faut jamais travailler sous tension, même s'il n'y a ici aucun danger pour le corps humain.

On commence par équiper le capteur (Figure 16.5) :

1. Prends un des trois straps, par exemple le rouge, et insère-le sur la broche du capteur correspondant au pôle positif. Sur certains modèles, c'est celui qui est à côté de la mention VCC, qui signifie Voltage Courant Continu. Dans d'autres cas, il y a le signe + ou encore la mention 5 V. Dans le modèle que j'ai préconisé, c'est la broche de gauche quand tu regardes le dos du circuit avec les broches en bas.
2. Insère ensuite le fil du signal sur la broche du milieu (souvent légendée OUT).
3. Insère enfin le fil noir sur la broche de la masse. La légende est souvent GND, qui signifie GrouND ou masse en anglais. C'est bien celle du 0 V.

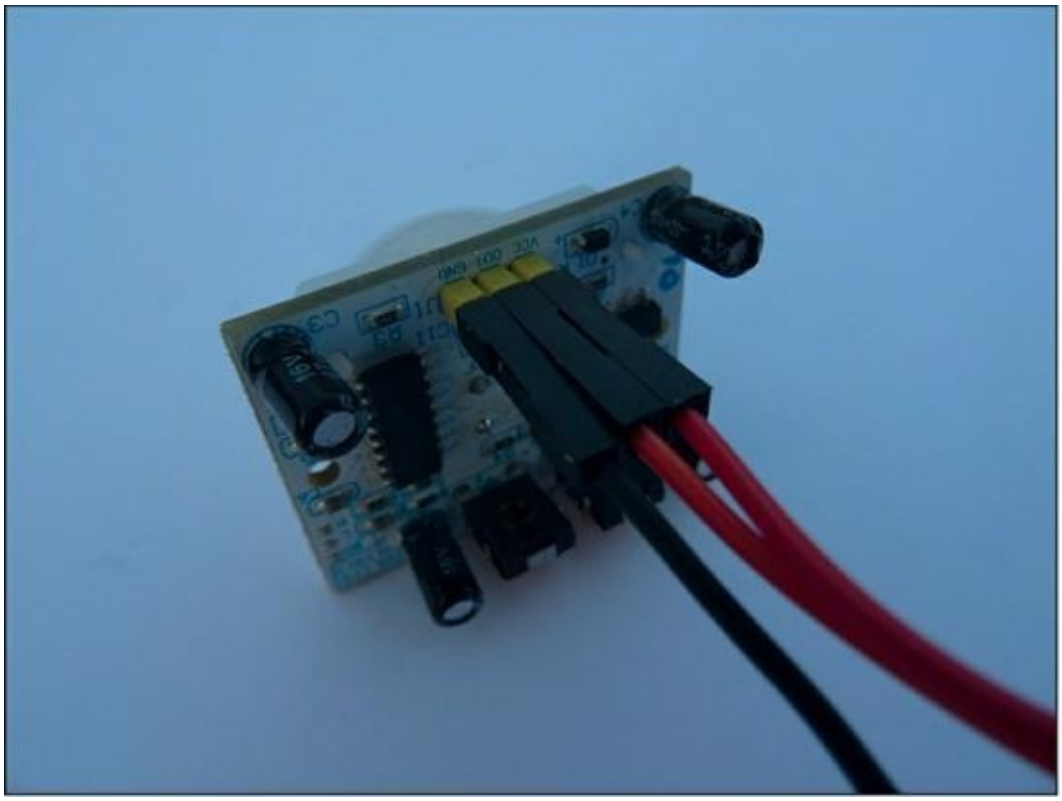


Figure 16.5 : Les trois broches du capteur équipées des straps.

Passons maintenant du côté du Raspi.

1. Si tu as trouvé des straps femelle/femelle, branche l'autre extrémité du fil rouge sur la broche du coin supérieur gauche du connecteur. Sur le schéma de la Figure 16.6, c'est la broche numéro 2.

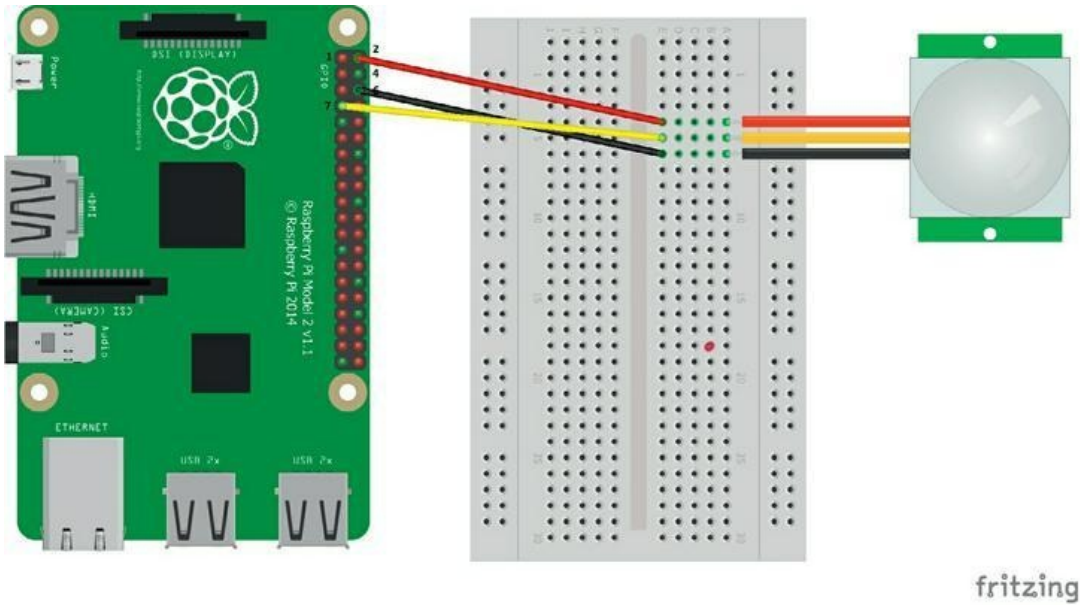


Figure 16.6 : Schéma de branchement sur le GPIO.

2. Insère maintenant le fil noir sur la broche de la masse, la broche numéro 6.
3. Insère enfin le fil du signal dans la broche numéro 7, la quatrième de la rangée du bas en partant de la gauche.

Le résultat doit ressembler à ce que montre la Figure 16.7.

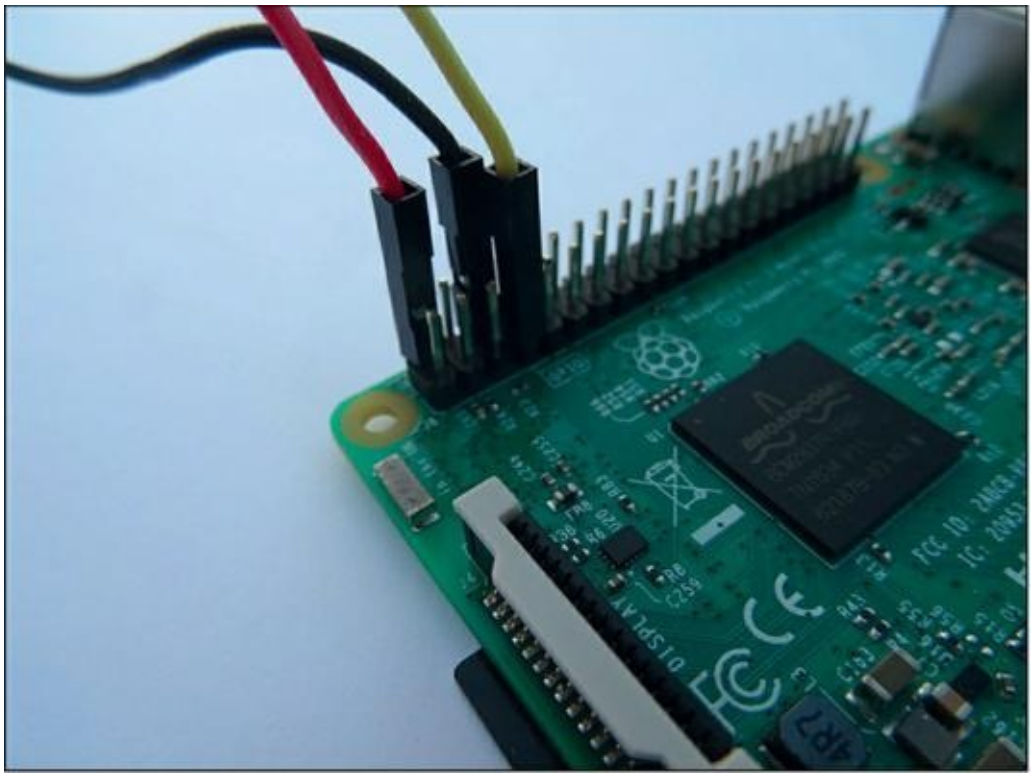


Figure 16.7 : Les trois straps branchés sur le GPIO.

Montage avec plaque d’essai

Si tu n’as pas trouvé de straps femelle/femelle, il te faut minimum des straps mâle/femelle. Dans ce cas, tu branches le côté femelle sur le capteur comme indiqué plus haut, puis tu insères les trois broches mâles sur une petite plaque d’essai (Figure 16.8).

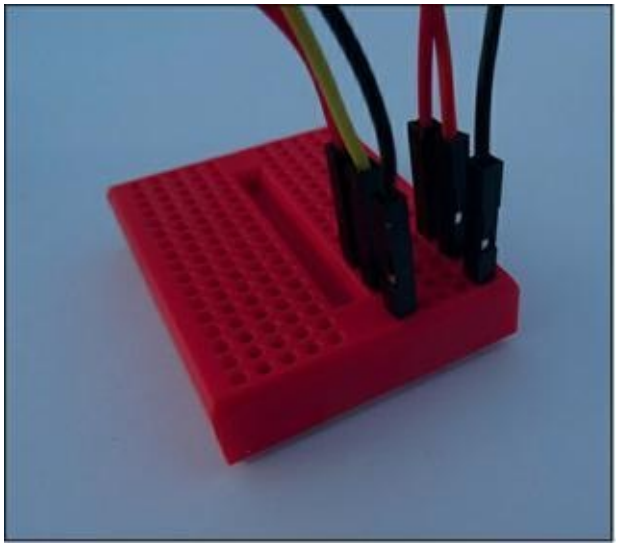


Figure 16.8 : Montage avec plaque d'essai intermédiaire.

Une plaque d'essai est une plaque comportant de nombreux trous arrangés en demi-rangées de contacts. La figure suivante montre comment sont interconnectés les deux straps de chaque couleur. Évidemment, tu dois disposer de trois autres straps mâle/femelle pour aller de la plaque d'essai vers le connecteur GPIO.

Mise sous tension

Avant de mettre sous-tension, revérifie que tu as branché les trois fils correctement, notamment au niveau des broches du connecteur GPIO. On peut facilement se perdre tellement les broches sont nombreuses et serrées.

Dorénavant, tu peux remettre ton Raspi sous tension. Il ne devrait rien se passer puisqu'il n'y a aucun programme qui se place à l'écoute de la broche numéro 7. Nous allons y remédier en écrivant un programme en langage Python.

Codage en Python

Notre programme doit s'adapter au mode de fonctionnement du capteur. Autrement dit, il doit être à tout moment dans un état parmi

les quatre suivants :

1. La phase de stabilisation du capteur est terminée et rien ne bouge dans le champ de vision.
2. La phase de stabilisation est terminée et quelque chose vient de bouger.
3. Quelque chose continue de bouger.
4. Plus rien ne bouge, mais la phase de restabilisation n'est pas terminée.

Après l'état 4, on finit nécessairement par revenir à l'état 1.

Pour écrire le programme, je te propose d'abord une découverte ligne par ligne, sans rien saisir dans l'éditeur de Python. Nous ferons cela dans un second temps.

Nous allons utiliser des fonctions qui sont déjà définies. Elles sont rassemblées dans une librairie de fonctions qui porte le nom `RPi.GPIO`. Fais bien attention au fait qu'il y a un i minuscule.

Nous allons également utiliser une autre librairie standard qui porte le nom `time` pour pouvoir utiliser une fonction pour mettre le circuit en attente pendant 10 ms pendant chaque tour de réarmement.

Nous commençons donc par demander l'importation de ces deux librairies :

```
import RPi.GPIO as GPIO
import time
```

Pour la première, nous abrégeons son nom ; ce qui suit le mot `as` est un synonyme.

Nous entrons ensuite dans le début du programme proprement dit, c'est-à-dire les instructions.

1) Configuration initiale

Du fait que le Raspi est très polyvalent, tu n'es pas étonné de voir qu'il nous faut choisir quelle convention de numérotation nous voulons utiliser pour indiquer le numéro de la broche du signal. On peut soit indiquer le numéro physique comme je l'ai dit plus haut (la broche numéro 7), soit le numéro tel que le connaît le microprocesseur. Comme par hasard, les deux numérotations ne coïncident pas. Voici comment s'écrit la fonction :

```
GPIO.setmode(GPIO.xxx)
```

La valeur que l'on indique à la place de `xxx` dans les parenthèses ne peut être que l'une des deux suivantes :

- `GPIO.BOARD` pour utiliser les numéros physiques des broches de la carte (carte se dit *board* en anglais) ;
- `GPIO.BCM` pour utiliser les numéros internes du processeur. Comme c'est un processeur de la marque Broadcom, le suffixe s'écrit BCM. Ces sont les numéros associés au sigle « GPIO » sur les schémas du connecteur. Ainsi, la broche physique N° 7 est en fait la broche GPIO4. (La broche GPIO7 correspond à la broche physique 26.)

Puisque nous utilisons la broche physique 7, nous utilisons l'option `GPIO. BOARD`. Voici donc comment s'écrit l'appel à la fonction pour choisir le mode de numérotation physique :

```
GPIO.setmode(GPIO.BOARD)
```

Pour simplifier la lecture de la suite du programme, nous allons déclarer une variable qui va symboliser le numéro de cette broche :

```
BrochePIR = 7
```

Ici, cette précaution n'est par d'un grand effet, puisque nous n'utilisons ce numéro qu'une fois dans la suite. Cela dit, c'est une bonne habitude à prendre.

La préparation n'est pas finie. Nous avons dit plus haut qu'il fallait pour chaque broche décider si on l'utilise en entrée ou en sortie. Voici l'appel de fonction correspondant :

```
GPIO.setup(BrochePIR, GPIO.IN)
```

C'est dans cette fonction que nous utilisons la variable que nous venons de définir comme étant équivalente à la valeur numérique 7. Le deuxième paramètre de la fonction est ici GPIO.IN. L'autre possibilité est GPIO.OUT, pour une sortie.

Il ne reste plus qu'à préparer deux variables dont nous aurons besoin sans arrêt dans la suite.

- Une première variable doit mémoriser l'état actuel qui peut être soit égal à 0, soit égal à 1.
- La seconde variable va mémoriser l'état antérieur qui pourra lui aussi être égal soit à 0, soit à 1.

L'état 1 signifie que quelque chose a été détecté et l'état 0 que c'est le calme plat. Voici la déclaration de ces variables :

```
EtatActuel = 0
EtatAvant  = 0
```

Nous donnons aux deux variables la valeur de départ 0, ce qui signifie qu'il ne vient pas d'y avoir détection et qu'il n'y en a pas eu avant. La section des préparatifs est ainsi terminée.

2) Phase de stabilisation initiale

Notre programme est prêt, mais le capteur peut-être pas encore. Pour lui laisser le temps de se préparer, nous prévoyons une boucle d'attente basée sur le mot-clé `while` :

```
while GPIO.input(BrochePIR) == 1 :
  EtatActuel = 0
```

Entre le mot-clé `while` et le signe deux-points, il y a une expression de test :

```
GPIO.input(BrochePIR) == 1
```

Dans cette expression, il y a deux membres séparés par un opérateur, ici celui de comparaison. Le membre de gauche est un appel de fonction.

```
GPIO.input(BrochePIR)
```

En écrivant cet appel, nous demandons d'exécuter une fonction qui porte le nom `input()` et qui appartient à la librairie `GPIO` (c'est l'alias que nous avons choisi). Nous lui indiquons quelle broche elle doit lire (`BrochePIR` contient la valeur 7).

Lorsque la fonction aura terminé, elle va renvoyer une valeur numérique. Si le capteur n'est pas encore prêt, ce sera un 1. Autrement dit, dans cette étape intermédiaire, la condition de la boucle sera devenue celle-ci :

```
while 1 == 1 :
```

Sans risque, je peux deviner que 1 est égal à 1, donc que l'expression est vraie, donc que le bloc conditionnel va être exécuté une fois. Ici le bloc commence après le signe deux-points et englobe toutes les lignes suivantes qui sont décalées vers la droite (ici, il n'y en a qu'une).

C'est le résultat final qui doit être vrai, c'est-à-dire différent de zéro. Nous aurions pu tester si 3 == 3 ; puisque c'est vrai, l'expression de `while` aurait été vraie.

Une fois cela fait, on revient à la première des deux lignes et on teste à nouveau la condition en commençant par lire la broche du signal, *etc*.

On ne quitte ces deux lignes que lorsque la broche d'entrée reçoit 0 V, ce qui prouve que le capteur est stabilisé. Dans ce cas, ce que renvoie la fonction ne sera plus égal à 1. Le capteur est alors prêt à repasser à 1 au prochain mouvement.

3) Boucle principale de surveillance

Nous entrons ensuite dans la partie répétitive du programme. Le capteur va sans cesse surveiller tout mouvement dans son champ de vision et se déclencher s'il détecte quelque chose puis passer en restabilisation dès que plus rien ne bouge.

Nous allons donc écrire tout un bloc qui sera répété sans cesse. Comment faire ? En utilisant à nouveau le mot-clé de condition TANT QUE, c'est-à-dire `while`, avec une expression à tester qui sera toujours vraie.

```
while True :
  première instruction conditionnelle
  première instruction conditionnelle
  ...
  dernière instruction conditionnelle
instruction normale après le bloc
```

Les lignes qui sont décalées par rapport à la marge font toutes partie du bloc conditionnel. Elles sont exécutées l'une après l'autre. Une fois la dernière exécutée, le processus reprend à la ligne de tête pour tester si la condition est toujours vraie. Ici, c'est une valeur fixe, le mot-clé `True`, donc on repart obligatoirement pour un tour de bloc.

Dès qu'une ligne ne commence plus avec un décalage par rapport à la tête `while`, on n'est plus dans le bloc. Dans notre exemple, elle ne

sera jamais exécutée, puisque la condition de la boucle `while` reste toujours vraie. Pour quitter le programme, il faudra se montrer brutal : frapper une combinaison de touches qui va forcer la fin d'exécution par le système. Ce n'est pas de cette façon qu'il faut procéder, mais nous allons nous en contenter pour cette première version.

En effet, il y a d'abord à comprendre la logique de la boucle, son algorithme.

Ce qui nous intéresse, c'est de savoir si la situation a changé. Au départ, après la phase de stabilisation, tout est calme. On consulte la broche sur laquelle est branché le fil de signal du capteur par un appel à la fonction `GPIO.input()` :

```
EtatActuel  =  GPIO.input(BrochePIR)
```

L'état actuel du capteur est mémorisé dans la variable d'état `EtatActuel` qui contient toujours le vrai état du capteur réel. L'état du capteur lors du tour de boucle précédent a été stocké dans l'autre variable d'état, `EtatAvant`.

En testant l'état actuel et l'état d'avant, on peut savoir ce qui se passe. Juste après la lecture du réel, on peut enchaîner nos tests :

Cas 1 : Alarme déclenchée !

```
if EtatActuel == 1 and EtatAvant == 0:
```

Traduction : SI Alarme déclenchée ET SI Stable au tour précédent => Capteur déclenché.

Dans ce cas, c'est que le fil de signal vient de changer d'état, donc on déclenche l'alarme. Surtout, on pense à stocker ce nouvel état d'alarme dans `EtatAvant` pour ne pas répéter inutilement le déclenchement pendant tout le temps où le capteur a besoin de se restabiliser. (Les prochaines lectures de la broche vont renvoyer la valeur 1, même si plus rien ne bouge.)

Cas 2 : Capteur restabilisé après alarme

```
elif EtatActuel == 0 and EtatAvant == 1:
```

Traduction : SI capteur stable ET on sort d'un état d'alarme => Capteur armé.

Dans ce cas, on vient de détecter que le capteur est redevenu stable, à l'état 0. Nous rebasculons donc en mode surveillance en forçant la variable `EtatAvant` à l'état calme.

Les deux autres cas possibles n'ont pas besoin d'un test spécifique. Ils sont pris en charge du simple fait que les deux premiers cas ne sont pas détectés.

Cas 3 : Capteur en cours de restabilisation après alarme

```
EtatActuel == 1 and EtatAvant == 1:
```

Traduction : SI Alarme en cours ET SI Alarme au tour précédent => Repartir pour un tour.

Après un déclenchement, le capteur maintient pendant quelques secondes les 3 V sur la broche de signal. Nous devons donc attendre que le signal redescende à 0. On ne peut rien faire d'autre qu'attendre, donc c'est reparti pour un tour de boucle.

En fait, on fait tout de même quelque chose : on marque un temps d'arrêt pour ne pas bombarder le capteur PIR d'interrogations de son état. La pause ne dure qu'un centième de seconde, mais pour le processeur, c'est beaucoup.

```
time.sleep(0.01)
```

Avec une vitesse d'horloge de 1 GHz, le processeur exécute environ un million d'instructions machine par seconde, soit dix mille

d’évitées pendant la durée de notre pause.

Cas 4 : Capteur stabilisé et en surveillance (armé)

```
EtatActuel == 0 and EtatAvant == 0:
```

Traduction : SI Calme ET SI Calme au tour précédent => Repartir pour un tour.

Su aucun mouvement n’a été détecté dans le tour actuel, ni dans le tour précédent, c’est que nous sommes dans l’état de surveillance. On marque une pause comme dans l’état 3 et on refait un tour de boucle.

Sortie de programme

Après la boucle, nous affichons un message et tentons d’appeler la fonction de mise en sécurité du connecteur GPIO. Tu constates que ces deux lignes ne sont plus indentées, à la différence des précédentes. Cela signifie qu’elles ne font pas partie du bloc de répétition :

```
GPIO.cleanup()
print “  OK. On quitte !”
```

La fonction `cleanup()` sert à forcer les broches GPIO qui ont été utilisées (ici, c’est la broche 7) en entrée, afin d’éviter tout court-circuit. Dans notre exemple, cela n’a aucun intérêt, mais c’est une précaution à connaître dès le départ.

Si tu avais utilisé la broche en sortie (avec `GPIO.setup(BrochePIR, GPIO. OUT)`) et que son dernier état soit l’état haut (3 V), le fait de faire toucher une masse par le fil qui y est branché aurait provoqué la destruction du circuit parce que cela aurait fait circuler trop de courant.

Saisie du code source

Grâce à ces explications détaillées, la saisie du code source devrait se dérouler sans problème, puisque tu dois maintenant deviner le but de chaque ligne.

1. Démarre l'éditeur Python par le sous-menu `Programmation`, puis `Python 2 (IDLE)`.

2. Dans la fenêtre Python 2.x Shell, ouvre le menu `File` et choisis `New File`.

3. Choisis immédiatement le nom du fichier de projet, avant même de commencer la saisie : ouvre le menu `File` et choisis `Save`. Donne au fichier le nom `proxi.py`.

4. Saisis le code source suivant en tenant compte des quelques remarques suivantes :

» Les lignes qui commencent par un signe dièse sont presque toutes des commentaires. Tu peux ne pas les saisir, sauf la première. Elle sert à pouvoir démarrer le programme depuis le terminal, en mode texte. La deuxième ligne qui cite le standard de codage des caractères `utf-8` garantit que tu peux utiliser des lettres accentuées du français dans les messages (mais jamais dans les noms de variables !).

» J'ai ajouté quelques lignes d'informations qui affichent des messages grâce à la fonction `print`. Dans cette première version, l'affichage d'un message est d'ailleurs la seule action faite quand un

mouvement est détecté. Pense aux guillemets ouvrants et fermants autour des messages à afficher.

- Attention aux indentations : dans la boucle `while` (sans compter les commentaires), quatre lignes d'instructions sont indentées d'un niveau (deux espaces) et quatre lignes de deux niveaux (quatre espaces).
- Dans les expressions de test, il y a bien à chaque fois deux signes égal (==) qui désignent l'opérateur de comparaison.
- Le signe deux-points est utilisé quatre fois en fin de ligne :

```
while GPIO.input(BrochePIR) == 1 :

while True  :

  if EtatActuel == 1 and EtatAvant == 0  :

  elif EtatActuel == 0 and EtatAvant == 1  :
```

Tu peux t'épargner la saisie de ce code source en chargeant le fichier `proxi. py` qui est fourni dans l'archive des exemples sur le site de l'éditeur (voir l'introduction), mais c'est moins formateur.

5. Enregistre ta saisie par le raccourci Ctrl-S.

Listing 16.1 : Code source de proxi.py.

```
#!/usr/bin/python
```

```
# -*- coding: utf-8 -*-
# Merci a Matt Hawkins de raspberrypi-
spy.co.uk
# Projet proxi.py

# Import des librairies de fonctions
import RPi.GPIO as GPIO
import time

# Num de broche physique
GPIO.setmode(GPIO.BOARD)

# On choisit la broche du signal sur le GPIO
BrochePIR = 7

print "*** Capteur de mouvements PROXI ***"
print "    Controle-C pour quitter."

# Broche en entrée
GPIO.setup(BrochePIR, GPIO.IN)

EtatActuel = 0
EtatAvant  = 0

print "Stabilisation du capteur..."
# On attend que le capteur se stabilise
while GPIO.input(BrochePIR) == 1 :
  EtatActuel = 0

print "On commence la surveillance."

# Boucle permanente (sortie par CTRL-C)
```

```
while True :

  # Lecture broche du capteur
  EtatActuel = GPIO.input(BrochePIR)

  if EtatActuel == 1 and EtatAvant == 0:
    # Le PIR a basculé
   print "  ALARME ! Quelque chose bouge !"
    # On se souvient de l’alarme
    EtatAvant = 1
  elif EtatActuel == 0 and EtatAvant == 1:
    # On vient de restabiliser
    print "  Restabilisation finie. On
traque..."
    EtatAvant = 0

  # On attend 10 millisecondes
  time.sleep(0.01)

# On nettoie le GPIO
GPIO.cleanup()
print “  OK. On quitte !”
# Fin du programme
```

La Figure 16.9 montre le même code source avec la colorisation appliquée par l’éditeur aux mots du langage Python.

Figure 16.9 : Le code source dans l'éditeur de Python 2.

Test du projet

Lance l'exécution du programme par la touche F5. (Si une erreur de compilation apparaît, reviens dans l'éditeur pour la corriger.)

Patiente quelques instants que le capteur soit stable. Déclenche un mouvement devant le capteur (tu peux aussi le secouer légèrement par ses fils). Un message d'alarme doit s'afficher, puis quelques

secondes plus tard, un autre pour confirmer que le capteur est à nouveau stable (Figure 16.10).

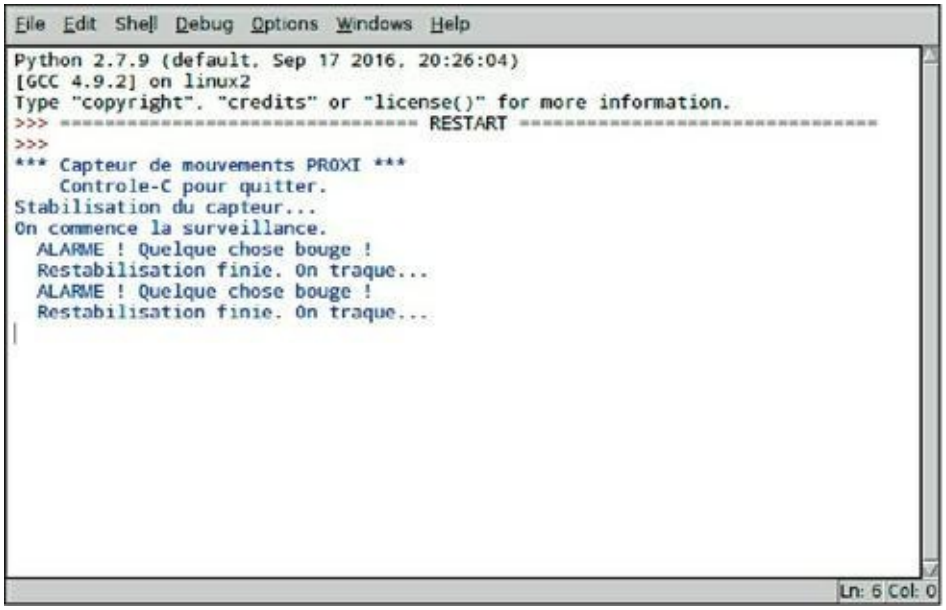


Figure 16.10 : Messages affichés pendant l'exécution.

En fonction du paysage présenté devant le capteur, le programme bascule entre les quatre états décrits plus haut.

Pas ou trop de détection ?

Il peut arriver que soit le capteur reste en état de détection, soit ne détecte jamais rien. Dans les deux cas, il suffit de régler la sensibilité. C'est ici que tu vas utiliser ton petit tournevis.

Sur le dos du capteur, tu vois deux potentiomètres (souvent de couleur orange). Repère celui marqué RP2 (ou bien regarde la notice). Introduis la lame du tournevis plat dans la fente (Figure 16.11) et tourne :

- dans le sens des aiguilles d'une montre (sens horaire) si le capteur ne détecte rien ;

> » dans le sens antihoraire, s'il ne veut pas revenir à l'état stable.

Tu vas trouver le bon réglage par tâtonnements successifs tout en regardant les messages apparaître.

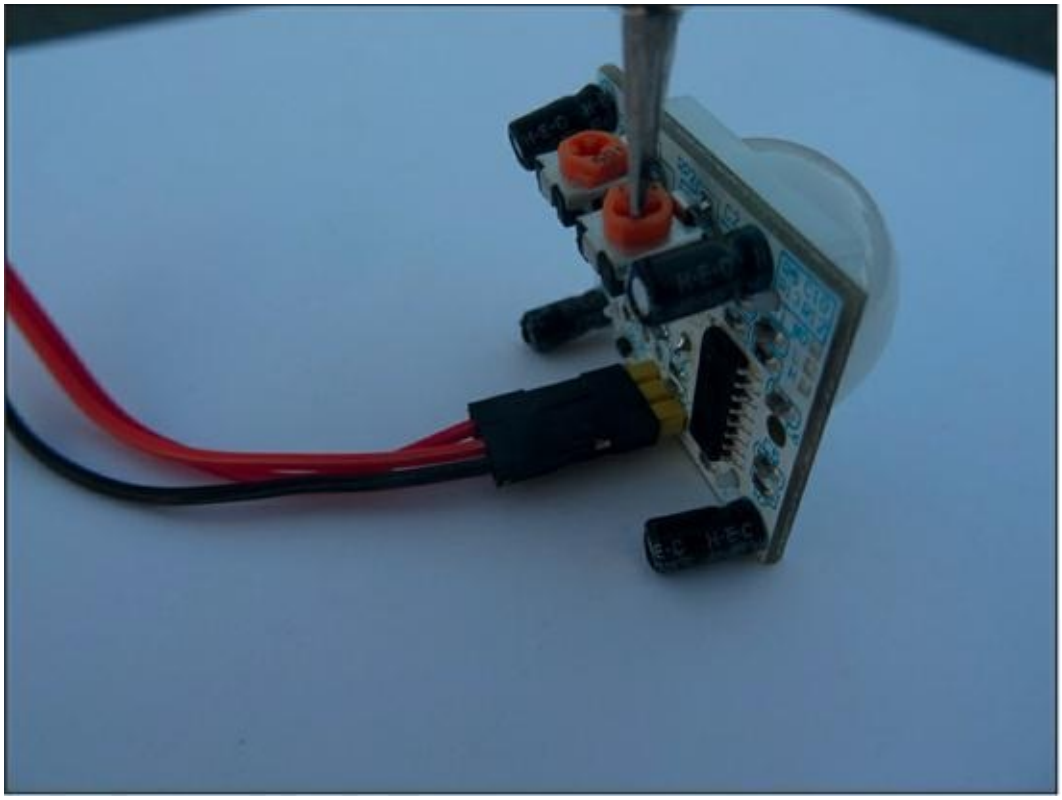


Figure 16.11 : Réglage du seuil de détection avec un tournevis.

Un arrêt d'exécution brutal

Pour quitter le programme, une seule solution : utiliser la séquence clavier magique Ctrl-C.

Le programme s'arrête, mais avec une méchante série de messages affichés en rouge dans la fenêtre d'exécution (Figure 16.12).

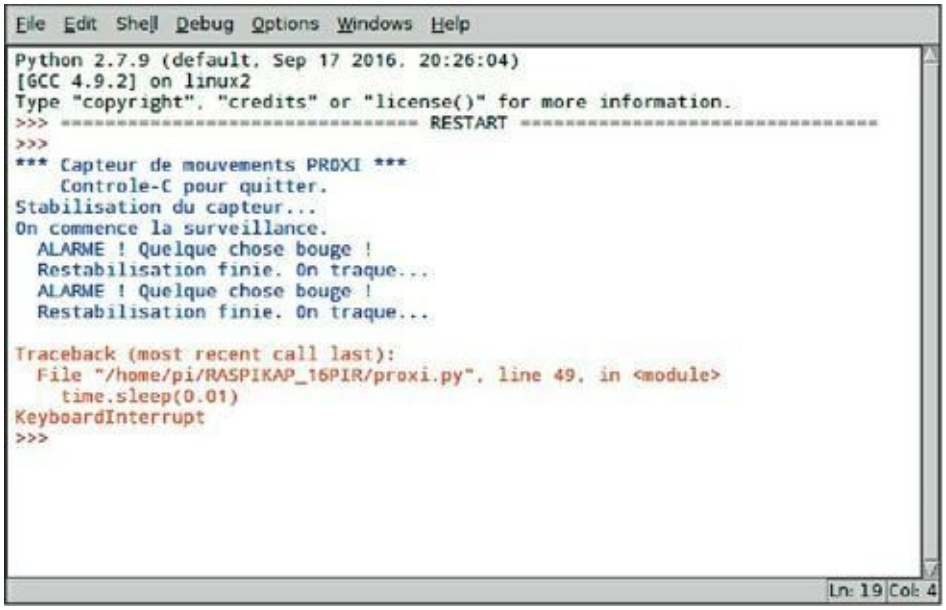


Figure 16.12 : Messages d'arrêt d'exécution brutal.

Est-ce grave, docteur ? Pas trop dans un petit projet comme celui-ci, mais ce n'est pas ainsi qu'il faut faire. En effet, tu as remarqué que l'instruction de mise en sécurité du connecteur n'a pas été exécutée, puisque le message « OK. On quitte ! » n'est pas apparu.

```
GPIO.cleanup()
print "  OK. On quitte !"
```

Quand le système Raspbian a détecté la frappe de Ctrl-C, il a sur-le-champ arrêté le programme qui était à ce moment dans la boucle. Il n'a donc pas pu en sortir pour poursuivre avec les deux instructions non indentées de la fin du code source.

Une bonne solution consiste à incorporer toute la boucle dans un bloc à exceptions qui commence par `try:` . Dans ce cas, le système redonne le contrôle à ton programme dans un bloc spécial de traitement de l'exception (qui commence par `except nom :`). Et la frappe de Ctrl-C est une exception.

Le principe d'un bloc try: except

Il n’y a que deux instructions à ajouter pour profiter de ce confort. Voici l’aspect général :

```
try:
  while True :
    instruction1
    instruction2
except NomException:
  instruction de sortie
```

Tu vois que toutes les lignes qui existaient déjà ont été décalées de deux espaces vers la droite. Pour en bénéficier :

1. Enregistre le code source actuel sous un nouveau nom, par exemple `proximi.py`.
2. Place-toi au début de la ligne de tête de la boucle permanente `while True :` et frappe Entrée pour insérer une ligne vide.
3. Saisis ceci (attention au signe deux-points) :

```
try:
```

4. Ajoute deux espaces au début de toutes les lignes suivantes, jusqu’à la fin. Le bloc conditionnel doit être dépendant de la ligne `try:` .

   Quand il y avait deux espaces, il en faut quatre. Quand il y en avait quatre, il en faut six. La moindre erreur va casser la logique du programme.

5. Place-toi au début de la ligne de l’instruction de nettoyage `cleanup()` et insère la ligne suivante, sans

espace au début !

`except KeyboardInterrupt :`

Dans le mot réservé du nom de l'interruption, seuls le K et le I sont en majuscules.

Tu dois obtenir le même résultat que dans le Listing 16.2. Ici, j'en ai profité pour enlever les précédents commentaires, puisque tu n'en as plus besoin. (En vrai, tu peux les laisser.)

Listing 16.2 : Code source de proximi.py.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Projet proximi.py

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
BrochePIR = 7

print "*** Capteur de mouvements PROXIMI
***"
print "    Controle-C pour quitter."

GPIO.setup(BrochePIR, GPIO.IN)

EtatActuel = 0
EtatAvant  = 0

print "Stabilisation du capteur..."
```

```
while GPIO.input(BrochePIR) == 1 :
  EtatActuel = 0

print "On commence la surveillance."

try:
  while True :
    EtatActuel = GPIO.input(BrochePIR)

    if EtatActuel == 1 and EtatAvant == 0:
      print "  Quelque chose bouge !"
      EtatAvant=1
    elif EtatActuel == 0 and EtatAvant == 1:
      print "  Restabilisation finie. On
traque..."
      EtatAvant = 0

    time.sleep(0.01)
except KeyboardInterrupt :
  # On nettoie le GPIO
  GPIO.cleanup()
  print "  OK. On quitte !"
# Fin du programme
```

Un arrêt d’exécution propre

Sauvegarde le programme et lance l’exécution par F5.

Comme auparavant, pour quitter le programme, tu utilises la séquence clavier magique Ctrl-C.

Cette fois-ci le programme s'arrête en douceur : il a le temps d'appeler la fonction de sécurité et d'afficher le message d'adieu (Figure 16.13).

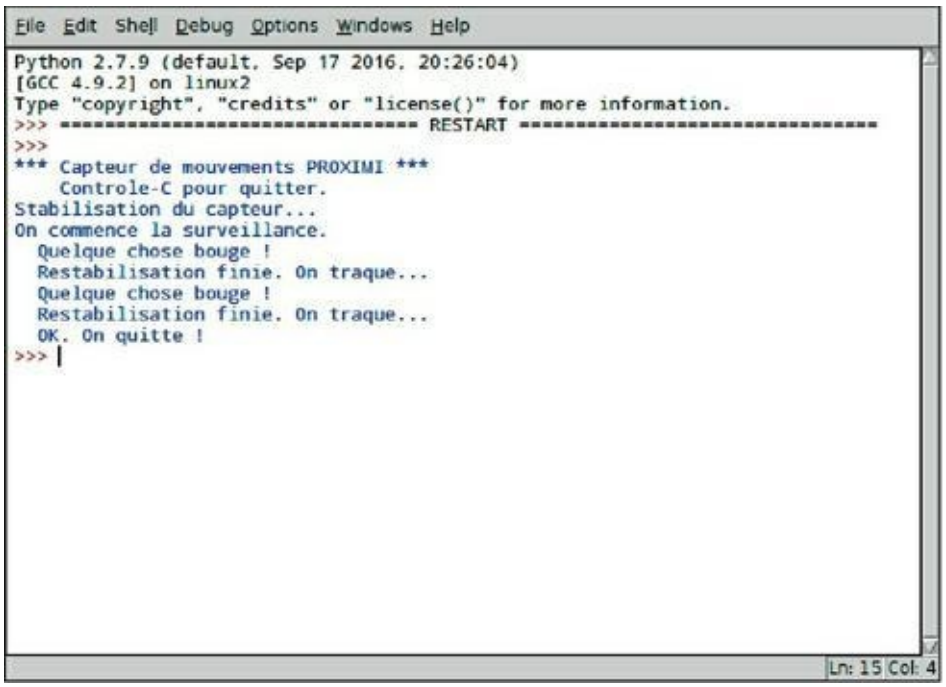


Figure 16.13 : Arrêt d'exécution contrôlé.

Nous pourrions nous arrêter ici, car tu as appris un certain nombre de techniques en quelques pages. Mais si tu as réalisé le montage de la caméra Raspberry Pi du chapitre précédent, tu peux aller plus loin : on peut prendre en photo ce qui a déclenché le capteur de mouvement ! Souriez, voleur, vous êtes filmé !

Le même avec l'image !

Si tu n'as pas installé la caméra officielle Raspberry Pi comme décrit dans le chapitre précédent, tu peux néanmoins lire la fin de ce chapitre pour découvrir une nouvelle technique relative aux fichiers uniques.

Puisque notre projet sait détecter un mouvement, tout est possible au niveau de la façon de réagir à cet événement. Il suffit d'ajouter des

instructions dans le sous-bloc conditionnel correspondant, celui qui affiche le message d'alarme.

Pour prendre une photo de ce qui a bougé devant l'objectif (à supposer que le capteur et la caméra visent dans la même direction, bien sûr), nous appelons la fonction `capture()` de la librairie `picamera`. Tu indiques comme paramètre d'appel le nom du fichier dans lequel va être stockée l'image :

```
maCam.capture(nomfic)
```

C'est la seule instruction à ajouter pour prendre la photo. Pour le nom de fichier, nous indiquons directement ce nom entre apostrophes (`'voleur. jpg'`).

Mais `maCam` est un objet ? Il faut donc d'abord créer cet objet, c'est-à-dire produire une instance de la classe `PiCamera`. Nous ajoutons donc une instruction en début de programme, juste après les directives d'import :

```
maCam = picamera.PiCamera()
```

Mais pour que le compilateur trouve la définition de la classe que nous citons, `PiCamera`, il faut lui préciser le nom de la librairie (`picamera`). On ajoute donc avant l'instruction de création de l'objet une directive d'import :

```
import picamera
```

Voici donc le code source complet de cette version avec prise de vue (Listing 16.3). Il n'y a que trois nouvelles lignes. Je les ai mises en valeur par un commentaire de fin de ligne `# AJOUT`.

Pour créer cette version, enregistre la version `proximi.py` sous le nom `proximicam.py` puis ajoute les trois lignes, enregistre et lance l'exécution.

Listing 16.3 : Code source de proximicam.py.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Projet proximicam.py

import RPi.GPIO as GPIO
import time

# Import pour picamera et objet maCam
import picamera                # AJOUT
maCam = picamera.PiCamera()    # AJOUT

GPIO.setmode(GPIO.BOARD)
BrochePIR = 7

print "*** Capteur de mouvements PROXIMICAM
***"
print "    Controle-C pour quitter."

GPIO.setup(BrochePIR, GPIO.IN)

EtatActuel = 0
EtatAvant  = 0

print "Stabilisation du capteur..."
while GPIO.input(BrochePIR) == 1 :
  EtatActuel = 0

print "On commence la surveillance."
```

```
try:
  while True :

    EtatActuel = GPIO.input(BrochePIR)

    if EtatActuel == 1 and EtatAvant == 0:
      print "  Quelque chose bouge !"
      EtatAvant=1

      # Souriez pour la photo
      maCam.capture('voleur.jpg')      #
AJOUT

    elif EtatActuel == 0 and EtatAvant == 1:
      print "  Restabilisation finie. On
traque..."
      EtatAvant = 0

    time.sleep(0.01)

except KeyboardInterrupt :
  # On nettoie le GPIO
  GPIO.cleanup()
  print "  OK. On quitte !"
# Fin du programme
```

Cette version ne permet de conserver que la dernière photo prise (Figure 16.14), puisque chaque nouvelle détection demande d'écrire l'image dans un fichier dont le nom est fixe.

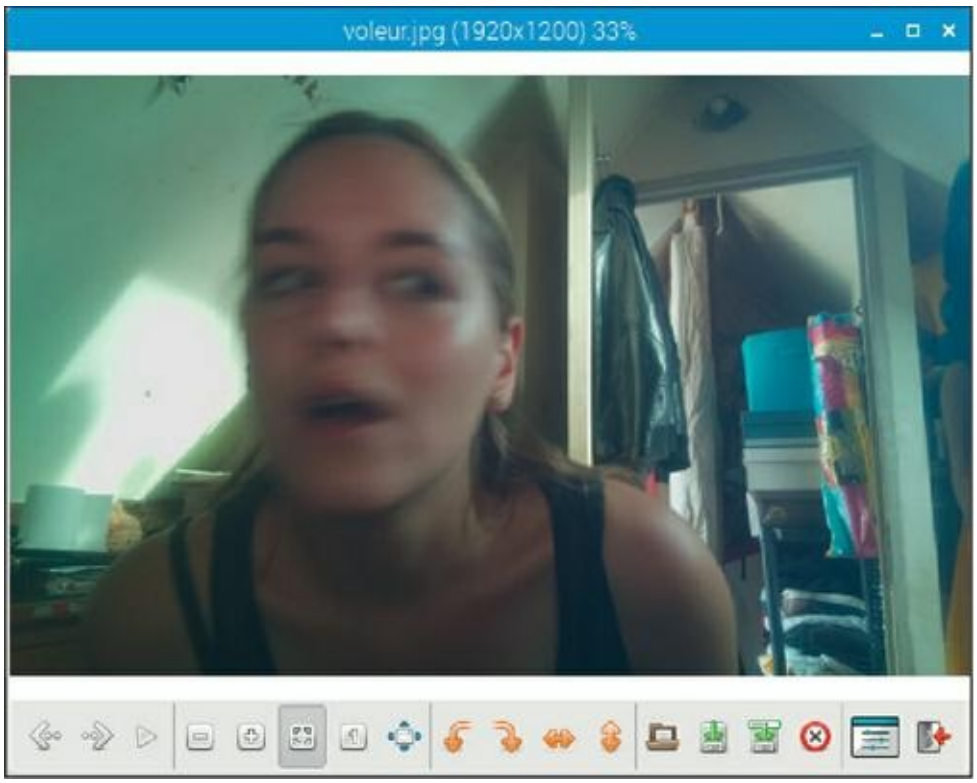


Figure 16.14 : Quelqu'un est passé devant le détecteur !

Comment faire pour conserver le cliché de chaque détection ? Il faudrait rendre le nom de fichier unique, donc fabriquer ce nom par exemple en y ajoutant un numéro.

De 1 à l'infini

Pour que chaque prise de vue n'efface pas la précédente par remplacement du contenu du fichier, nous pourrions utiliser une fonction Python de génération d'un nombre au hasard (`random()`). Le résultat serait une litanie de noms de fichiers sans aucun pouvoir de suggestion.

Quelle valeur du monde réel, et même de tout l'univers depuis ses origines, est par définition unique ? La date et l'heure, bien sûr ! Chaque seconde de notre vie est unique (tu le savais déjà, mais on l'oublie souvent).

Est-ce que le langage Python possède une fonction qui pourrait nous aider ? Peut-être dans la librairie standard justement nommée `time` ?

Ouiiiii ! Il y a une fonction nommée `strftime()` qui va lire l'heure du système et la renvoyer dans le format que nous lui demandons en entrée par une chaîne de format.

```
time.strftime("%Y%m%dh%H%M%S")
```

Cette chaîne est un peu bizarre au premier regard, mais il suffit de la décortiquer :

%Y	L'année (Year) sur quatre chiffres
%m	Le mois sur deux chiffres
%d	Le jour sur deux chiffres
h	La lettre h telle quelle (littérale)
%H	L'heure sur deux chiffres
%M	La minute sur deux chiffres
%S	La seconde sur deux chiffres

J'ai ajouté la lettre `h` minuscule pour séparer la partie jour de la partie heure. J'aurais pu demander une espace, mais je déconseille fortement d'utiliser l'espace dans les noms de fichiers. En mode texte dans le terminal et dans les scripts, c'est la galère assurée.

Le nom de la fonction signifie *STring Format Time*, une chaîne de caractères d'heure formatée.

Cette fonction va donc renvoyer une chaîne horodatrice. Il faut penser à la récupérer dans une variable. Nous utilisons à cet effet la variable `timestr` dont nous affichons la valeur pour contrôle.

```
timestr = time.strftime("%Y%m%dh%H%M%S")
print(timestr)
```

Nous pourrions maintenant utiliser directement cette chaîne comme nom du fichier image à sauvegarder :

```
maCam.capture(timestr)
```

Mais puisque nous pouvons construire une nouvelle chaîne à partir de deux autres par l'opérateur plus, autant ajouter une mention à la suite de la date :

```
nomfic = timestr + '_detection.jpg'
```

La nouvelle chaîne est `nomfic`. C'est elle que nous transmettons à la fonction de capture de la photo, et le tour est joué :

```
maCam.capture(nomfic)
```

Pour la partie personnalisée entre apostrophes, tu peux choisir ce que tu veux, mais pas d'accents, pas de signes bizarres, pas d'espace.

Voici donc la dernière version de notre projet. J'ai enlevé dans le Listing 16.4 tout le début qui ne change pas puisque la librairie `time` était déjà demandée pour notre pause de fin de tour de boucle.

Saisis les nouvelles lignes et modifie celle d'appel à `capture()` puis enregistre sous un autre nom et lance l'exécution.

Listing 16.4 : Code source partiel de proximicamdate.py.

```
# Code inchangé jusqu'à la boucle principale
try:
#...

try:
```

```
  while True :

    EtatActuel = GPIO.input(BrochePIR)

    if EtatActuel == 1 and EtatAvant == 0:
      print «  Quelque chose bouge ! »
      EtatAvant=1

      # On fabrique la chaine horodatrice
      timestr =
time.strftime(«%Y%m%dh%H%M%S»)  # AJOUT

      print(timestr)
# AJOUT
      # On utilise la chaine horodatrice
      nomfic = timestr + ‘_detection.jpg’
# AJOUT

      maCam.capture(nomfic)
# MODIF
    elif EtatActuel == 0 and EtatAvant == 1:
      print «  Restabilisation finie. On
traque...»
      EtatAvant = 0

    time.sleep(0.01)

except KeyboardInterrupt :
  GPIO.cleanup()
  print “  OK. On quitte !”
# Fin du programme
```

En ouvrant une fenêtre du Gestionnaire de fichiers puis en allant dans le sous-répertoire contenant le code source, tu dois constater l'apparition d'au moins un fichier horodaté. Cet horodatage est très pratique puisque les fichiers sont d'office triés en ordre chronologique si tu choisis l'ordre `Par nom` dans le menu `Voir/Trier les fichiers`.

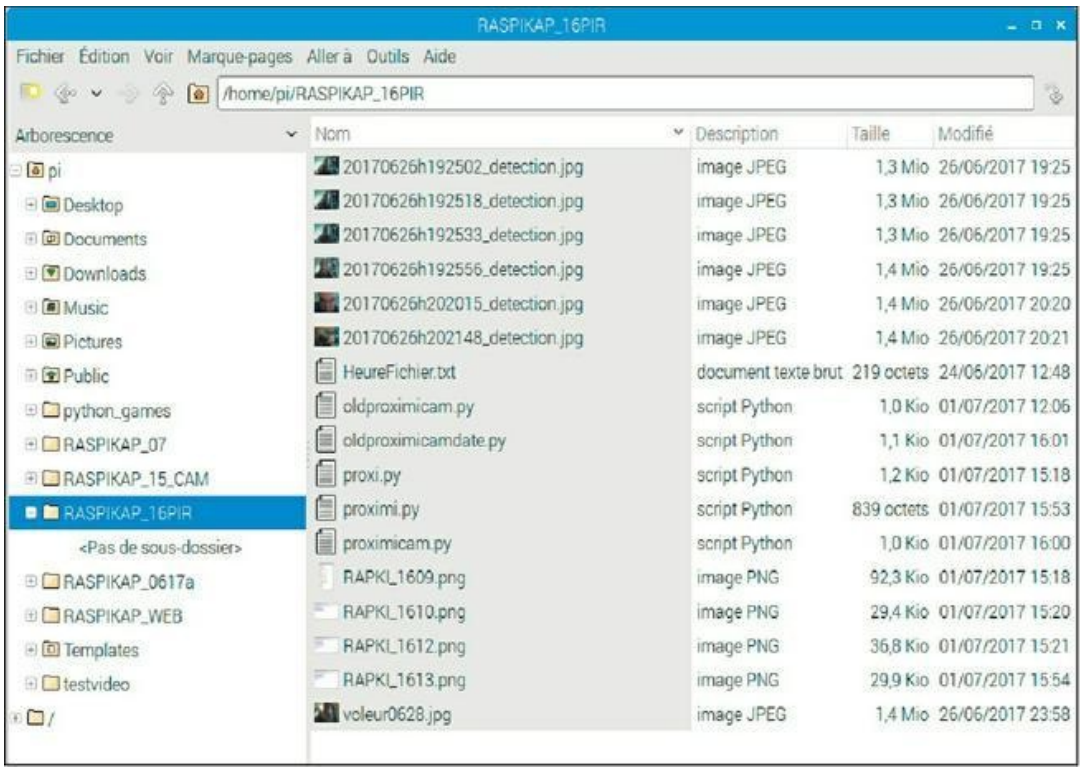


Figure 16.15 : Plusieurs fichiers générés avec un horodateur (début de liste).

Conclusion

Apparemment, nous sommes arrivés au bout du voyage.

En réalité, ce qui commence maintenant, c'est ton grand voyage personnel dans le monde de la cybernétique. Raspberry Pi est un circuit fantastique, car il rapproche deux domaines souvent distincts : l'informatique de type bureautique et ludique, celle des ordinateurs

personnels en réseau, et l'informatique scientifique, celle des capteurs, des robots et des drones.

Pour poursuivre avec un guide, je te propose chez le même éditeur :

- *Programmer pour les Nuls* (Olivier Engler et Wallace Wang)
- *Programmer avec Python en s'amusant* (Brendan Scott)
- *Programmer avec Arduino en s'amusant pour les Nuls* (Olivier Engler).

Bon voyage !

Sommaire