EYROLLES

RAPHAËL **HERTZOG**
& ROLAND **MAS**
Ouvrage dirigé par NAT **MAKARÉVITCH**

GNU/Linux Debian

iSBN : 978-2-212-12505-4

EYROLLES

annexe A


Bitstream Vera
Courrier entrant pour
Affichage
Aller à
Messages
Adresses
Voir :
Tous
Sujet
Compte personnel
Courrier entrant
Dossiers locaux
Corbeille
Edition
Enigmail
Outils
Aide
Relever
Écrire
Adresses
Dossiers
Le sujet ou l'expéditeur contient
Compte personnel

Distributions dérivées

De nombreuses distributions dérivent de Debian et emploient ses outils. Chacune présente des particularités intéressantes et comblera peut-être vos attentes !

SOMMAIRE

MOTS-CLÉS

- Live CD
- Spécificités
- Choix particuliers

Ubuntu Linux

Ubuntu Linux est une des plus récentes distributions dérivées de Debian, mais son entrée sur la scène du logiciel libre a été très médiatique. Et pour cause : la société Canonical Ltd. qui a créé cette distribution a embauché une trentaine de développeurs Debian en affichant l'ambitieux objectif de faire une distribution pour le grand public, et de publier une nouvelle version tous les 6 mois. Ils promettent par ailleurs de maintenir chaque version pendant 18 mois.

Pour parvenir à leurs objectifs, ils se concentrent sur un nombre restreint de logiciels, et s'appuient essentiellement sur GNOME. Tout est internationalisé et disponible dans un grand nombre de langues, dont le français.

Force est de constater que, pour le moment, ils arrivent à maintenir ce rythme de publication. Ils ont en outre introduit le concept d'une version *Long Term Support* (LTS) qui est supportée sur 3 ans pour la partie bureautique et sur 5 ans pour la partie serveur. La distribution Dapper Drake (6.06) est la version LTS courante tandis que Gutsy Gibbon (7.10) est la dernière version stable. Les numéros de versions symbolisent simplement la date de publication : 7.10 représente le mois d'octobre 2007.

A contrario, si le succès d'Ubuntu est évident auprès du grand public, tout n'est pas aussi rose pour les développeurs Debian qui espéraient beaucoup d'Ubuntu en termes d'améliorations directes apportées à Debian. Le marketing de Canonical laisse croire qu'ils sont de bons citoyens du logiciel libre, simplement parce qu'ils mettent à disposition les changements effectués dans les paquets Debian. En réalité ces patches sont générés automatiquement et contiennent fréquemment plusieurs changements entremêlés, de telle sorte qu'un développeur Debian aura énormément de mal à y récupérer uniquement le changement qui l'intéresse. En outre, un bon citoyen du logiciel libre envoie ses modifications avec des explications qui justifient les différents changements et interagit avec le mainteneur de sorte à obtenir le meilleur résultat final. Chez Ubuntu, il n'existe pas une telle volonté politique : tout repose sur la bonne volonté de chacun des employés et contributeurs. Certains n'ont pas oublié leurs racines et font l'effort nécessaire (citons notamment Colin Watson, Martin Pitt et Matthias Klose), mais d'autres — souvent surchargés par le travail — n'arrivent plus à trouver la volonté nécessaire.

▸ http://www.ubuntu.com/

Knoppix

La distribution Knoppix n'a presque plus besoin d'être présentée. Elle a popularisé le concept de *LiveCD* : il s'agit d'un CD-Rom amorçable qui démarre directement un système Linux fonctionnel et prêt à l'emploi, sans nécessiter de disque dur — tout système déjà présent sur la machine sera donc laissé intact. L'autodétection des périphériques permet à cette distribution de fonctionner avec presque toutes les configurations matérielles. Le CD-Rom contient près de 2 Go de logiciels compressés.

Si vous cumulez ce CD-Rom avec une clé USB, vous pourrez emmener vos fichiers avec vous et travailler sur n'importe quel ordinateur sans laisser de trace — rappelons que la distribution n'utilise pas du tout le disque dur. Knoppix est essentiellement fondé sur KDE, mais de nombreuses distributions dérivées proposent d'autres combinaisons de logiciels. Citons notamment Morphix, qui s'appuie sur une structure modulaire permettant d'offrir plusieurs LiveCD — chacun avec sa propre sélection de logiciels.

Signalons en outre que la distribution offre malgré tout un installateur : vous pourrez ainsi essayer Knoppix en tant que *LiveCD* puis, une fois convaincu, l'installer sur le disque dur pour obtenir de meilleures performances.

- http://www.knoppix-fr.org/
- http://www.morphix.org/

Mepis Linux

Mepis Linux est une distribution commerciale très similaire à Knoppix. Proposant un système Linux prêt à l'emploi depuis un *LiveCD*, cette distribution intègre un certain nombre de logiciels qui ne sont pas libres : les pilotes pour les cartes graphiques nVidia, Macromedia Flash pour les animations intégrées à de nombreux sites web, RealPlayer, le Java de Sun, etc. L'objectif est d'offrir un système 100 % fonctionnel dès l'installation. Mepis est internationalisée et gère la langue française.

- http://www.mepis.org/

Cette distribution initialement basée sur Debian est désormais basée sur Ubuntu. Cela lui permet de se concentrer sur l'ajout de fonctionnalités sans devoir s'occuper de stabiliser les paquets récupérés depuis la distribution *Unstable* de Debian.

Xandros

Xandros Linux est une distribution commerciale traditionnelle, qui dispose d'un installateur graphique ultra-simplifié en 4 étapes. Elle cible principalement les utilisateurs anglophones, puisque seul l'anglais est

disponible dans l'installateur et dans le manuel inclus dans la version « boîte ». Comme tout système Linux, il est toutefois toujours possible de configurer le système en français après l'installation.

Il existe plusieurs versions de la distribution, adaptées à des usages différents : la version familiale propose un système standard adapté pour un usage bureautique et ludique ; la version entreprise propose en plus des outils de gestion de parc de machines, etc.

- http://www.xandros.com/

Linspire et Freespire

Cette distribution commerciale vise le grand public, la société éditrice passe des accords de distribution OEM importants aux États-Unis (notamment avec le distributeur Walmart). Son dirigeant principal, Michael Robertson, est connu pour ses prises de position très tranchées. Le premier nom de la distribution était d'ailleurs « Lindows », et une longue guerre juridique les a opposés à Microsoft parce que ce nom était trop proche de celui de leur système d'exploitation Windows.

- http://www.linspire.com/

Les dernières versions sont basées sur Ubuntu et permettent d'y intégrer des codecs, pilotes et applications propriétaires selon les besoins de l'utilisateur.

- http://www.freespire.org/

Une version communautaire de la distribution existe, elle s'appelle Freespire.

Damn Small Linux

- http://www.damnsmalllinux.org/

Cette distribution propose un *LiveCD* de 50 Mo afin qu'il tienne sur un CD-Rom ayant la forme et la taille d'une carte de visite (*businesscard CD*). Cela peut être intéressant pour utiliser un système Debian sur un vieil ordinateur.

Et d'autres encore

- http://distrowatch.com

Le site Distrowatch référence de très nombreuses distributions Linux, dont un grand nombre sont basées sur Debian. N'hésitez pas à le parcourir pour constater la diversité du monde du logiciel libre !

Le formulaire de recherche permet de retrouver les distributions en fonction de celle sur laquelle elles se basent. En sélectionnant Debian, le formulaire ne renvoie pas moins de 132 distributions actives !

annexe B

Petit cours de rattrapage

Bien que cet ouvrage s'adresse principalement aux administrateurs ou utilisateurs avancés, nous ne souhaitons pas exclure les novices désireux de se former. Nous rappelons donc dans cette partie quelques concepts informatiques fondamentaux que l'on côtoie en exploitant un ordinateur sous Unix.

SOMMAIRE

MOTS-CLÉS

- BIOS
- Noyau
- Unix
- Processus
- Arborescence
- Commandes de base

Interpréteur de commandes et commandes de base

Dans le monde Unix, l'administrateur est inévitablement confronté à la ligne de commande, ne serait-ce que dans les cas où le système ne démarre plus correctement et qu'il propose juste un accès de secours en ligne de commande. Il est donc important de savoir se débrouiller un minimum dans un interpréteur de commandes.

Les commandes présentées dans cette section le sont de manière assez rapide, il ne faut pas hésiter à consulter les pages de manuels correspondantes pour découvrir les nombreuses options disponibles.

Déplacement dans l'arborescence et gestion des fichiers

Après connexion, la commande **pwd** (*print working directory* soit « afficher le répertoire de travail ») permet de connaître l'emplacement courant. Pour changer de répertoire courant, c'est la commande **cd *répertoire*** (*change directory* soit « changer de répertoire ») qui le permet. Le répertoire parent est toujours nommé `..` tandis que `.` est un synonyme pour le répertoire courant. La commande **ls** permet d'afficher le contenu d'un répertoire, en l'absence de paramètres elle travaille sur le répertoire courant.

```
$ pwd
/home/rhertzog
$ cd Desktop
$ pwd
/home/rhertzog/Desktop
$ cd .
$ pwd
/home/rhertzog/Desktop
$ cd ..
$ pwd
/home/rhertzog
$ ls
Desktop  Mail  tmp
```

Créer un nouveau répertoire s'effectue avec **mkdir *répertoire*** alors que la commande **rmdir *répertoire*** permet de supprimer un répertoire vide. La commande **mv** permet de renommer et/ou de déplacer les fichiers et les répertoires, tandis que **rm *fichier*** permet de supprimer un fichier.

```
$ mkdir test
$ ls
Desktop  Mail  test  tmp
$ mv test nouveau
$ ls
```

```
Desktop  Mail  nouveau  tmp
$ rmdir nouveau
$ ls
Desktop  Mail  tmp
```

Consultation et modification des fichiers texte

La commande **`cat` *`fichier`*** (prévue pour concaténer des fichiers sur la sortie standard) permet simplement de lire un fichier et d'afficher son contenu dans le terminal. Si le fichier est trop gros, la commande **`less`** (ou **`more`**) permet de l'afficher page par page.

La commande **`editor`** pointe toujours sur un éditeur de texte (comme **`vi`** ou **`nano`**) et permet de créer/modifier/lire des fichiers textes. Pour les fichiers les plus simples, il est parfois possible de les créer directement depuis l'interpréteur de commandes grâce aux redirections. Ainsi **`echo "`*`texte`*`" >`*`fichier`** crée un fichier nommé *fichier* contenant « *texte* ». Pour rajouter une ligne à la fin de ce fichier, il est possible de faire **`echo "`*`ligne`*`" >>`*`fichier`*.

Recherche de fichiers et dans les fichiers

La commande **`find` *`répertoire critères`*** permet de rechercher des fichiers dans l'arborescence sous *répertoire*. L'option `-name` *`nom`* est le critère de recherche le plus courant et permet de retrouver un fichier par son nom.

La commande **`grep` *`expression fichiers`*** permet de rechercher le contenu des fichiers et d'en extraire les lignes correspondant à l'expression rationnelle (voir encadré page 223). L'option `-r` permet de faire une recherche récursive sur tous les fichiers contenus dans le répertoire indiqué en paramètre. Cela permet d'identifier facilement un fichier dont on connaît une partie du contenu.

Gestion des processus

La commande **`ps aux`** permet de consulter la liste des processus en cours d'exécution et de les identifier par leur *pid*. Par la suite, la commande **`kill -`*`signal pid`*** permet d'envoyer un signal à un processus donné (à condition qu'il s'agisse d'un processus du même utilisateur). Le signal `TERM` demande au programme de se terminer alors que `KILL` le tue brutalement.

L'interpréteur de commandes permet de lancer des programmes en tâche de fond : il suffit pour cela de rajouter « & » à la fin de la commande. Dans ce cas, l'utilisateur retrouve le contrôle immédiatement bien que la commande lancée ne soit pas encore terminée. La commande **`jobs`** permet d'identifier les processus exécutés en arrière-plan. La commande **`fg %`*`numéro-de-job`*** (*foreground* soit « avant-plan ») replace le processus à

l'avant-plan. Dans cette situation, la combinaison de touche *Control* + *Z* permet de stopper l'exécution du processus et de reprendre le contrôle de la ligne de commande. Pour réactiver en arrière-plan le processus stoppé, il faut faire **bg %*numéro-de-job***.

Informations système : mémoire, espace disque, identité

La commande **free** affiche des informations sur l'usage de la mémoire vive, tandis que **df** (*disk free*) rapporte l'espace disponible sur les différents disques accessibles dans l'arborensce. On emploie fréquemment l'option `-h` de **df** (pour *human readable*) afin qu'il affiche les tailles avec une unité plus adaptée (généralement mégaoctets ou gigaoctets). De même, la commande **free** dispose de `-m` ou `-g` pour afficher les informations soit en mégaoctets soit en gigaoctets.

```
$ free
             total       used       free     shared    buffers     cached
Mem:       1028420    1009624      18796          0      47404     391804
-/+ buffers/cache:     570416     458004
Swap:      2771172     404588    2366584
$ df
Sys. de fich.        1K-blocs       Occupé Disponible Capacité Monté sur
/dev/hda6              9614084   4737916   4387796  52% /
tmpfs                   514208         0    514208   0% /lib/init/rw
udev                     10240       100     10140   1% /dev
tmpfs                   514208    269136    245072  53% /dev/shm
/dev/hda7             44552904  36315896   7784380  83% /home
```

La commande **id** permet d'afficher l'identité de l'utilisateur connecté et indique la liste des groupes dont il est membre. Il est parfois important de pouvoir vérifier si l'on est membre d'un groupe donné, cela peut conditionner l'accès à certains fichiers ou périphériques.

```
$ id
uid=1000(rhertzog) gid=1000(rhertzog) groupes=20(dialout),24(cdrom),
  ➥ 25(floppy),29(audio),44(video),46(plugdev),105(netdev),
  ➥ 116(vde2-net),1000(rhertzog)
```

Organisation de l'arborescence des fichiers

La racine

L'arborescence d'un système Debian est organisée selon la norme FHS (*File Hierarchy Standard*). Elle codifie de manière précise l'usage de chaque répertoire. Étudions la subdivision principale :

- `/bin/` : programmes de base ;
- `/boot/` : noyau Linux et autres fichiers nécessaires à son démarrage ;
- `/dev/` : fichiers de périphériques ;
- `/etc/` : fichiers de configuration ;
- `/home/` : fichiers personnels des utilisateurs ;
- `/lib/` : bibliothèques de base ;
- `/media/*` : points de montage pour des périphériques amovibles (CD-Rom, clé USB, etc.) ;
- `/mnt/` : point de montage temporaire ;
- `/opt/` : applications additionnelles fournies par des tierces parties ;
- `/root/` : fichiers personnels de l'administrateur (utilisateur root) ;
- `/sbin/` : programmes systèmes ;
- `/srv/` : données pour les services hébergés par ce système ;
- `/tmp/` : fichiers temporaires, ce répertoire étant souvent vidé au démarrage ;
- `/usr/` : applications supplémentaires ; ce répertoire se subdivise à nouveau en `bin`, `sbin`, `lib` selon la même logique. En outre `/usr/share/` contient des données indépendantes de l'architecture. `/usr/local/` permet à l'administrateur d'installer manuellement certaines applications sans perturber le reste du système qui est géré par le système de paquetage (**`dpkg`**).
- `/var/` : données variables des démons. Ceci inclut les fichiers de traces, les files d'attente, les caches, etc.
- `/proc/` et `/sys/` ne sont pas standardisés et sont spécifiques au noyau Linux. Ils servent à exporter des données du noyau vers l'espace utilisateur.

Le répertoire personnel de l'utilisateur

Le contenu des répertoires utilisateurs n'est pas standardisé, il n'empêche qu'il y a tout de même quelques conventions à connaître. Mais avant tout il faut savoir que l'on désigne fréquemment le répertoire personnel par un tilde (« ~ ») car les interpréteurs de commande le remplaceront automatiquement par le bon répertoire `/home/`*`utilisateur/`*.

Les fichiers de configuration des applications sont directement dans le répertoire de l'utilisateur mais leurs noms débutent par un point (ex : `~/.muttrc` pour le lecteur de courrier **`mutt`**). Les fichiers débutant par un point sont cachés par défaut : il faut passer l'option `-a` à **`ls`** pour les voir.

Parfois, les logiciels emploient un répertoire complet (comme `~/.evolution/`) lorsqu'ils ont plusieurs fichiers de configuration à

stocker. Signalons au passage que certaines applications (les navigateurs web comme Iceweasel par exemple) utilisent ces répertoires comme cache pour des données téléchargées. C'est pourquoi certains de ces répertoires peuvent être assez gros.

Les bureaux graphiques affichent le contenu du répertoire `~/Desktop/` sur le bureau (c'est l'écran qui reste une fois toutes les applications fermées ou minimisées).

Enfin, il arrive que le système de messagerie dépose les courriers électroniques entrants dans `~/Mail/`.

Fonctionnement d'un ordinateur : les différentes couches en jeu

L'ordinateur se présente souvent comme quelque chose d'assez abstrait, et sa partie visible est très simplifiée par rapport à sa complexité réelle. Cette complexité réside en partie dans le nombre d'éléments mis en jeu ; ces éléments peuvent cependant être regroupés en couches superposées les éléments d'une couche n'interagissant qu'avec ceux de la couche immédiatement supérieure et de la couche immédiatement inférieure.

En tant qu'utilisateur final, il n'est pas forcément nécessaire de connaître ces détails... du moins tant que tout fonctionne. Une fois confronté au problème « l'accès Internet ne fonctionne plus », il est indispensable de pouvoir retrouver dans quelle couche le problème apparaît : est-ce que la carte réseau (le matériel) fonctionne ? Est-ce qu'elle est reconnue par l'ordinateur ? Est-ce que Linux la reconnaît ? Est-ce que le réseau est bien configuré, etc. Toutes ces questions vont permettre d'isoler la couche responsable et de traiter le problème au bon niveau.

Au plus bas niveau : le matériel

Pour commencer, rappelons qu'un ordinateur est avant tout un ensemble d'éléments matériels. On a généralement une carte-mère, sur laquelle sont connectés un processeur (parfois plusieurs), de la mémoire vive, différents contrôleurs de périphériques intégrés, et des emplacements d'extension pour des cartes filles, pour d'autres contrôleurs de périphériques. Parmi ces contrôleurs, on peut citer les normes IDE (Parallel ATA), SCSI et Serial ATA, qui permettent de raccorder des périphériques de stockage comme des disques durs. On trouve également des contrôleurs USB, qui accueillent une grande variété de matériels (de la webcam au thermomètre, du clavier à la centrale domotique) et IEEE 1394 (Firewire). Ces contrô-

leurs permettent souvent de relier plusieurs périphériques à la fois, c'est pourquoi on emploie fréquemment le terme de « bus » pour désigner le sous-système complet géré par le contrôleur. Les cartes filles incluent les cartes graphiques (sur lesquelles on pourra brancher un écran), les cartes son, les cartes réseau, etc. Certaines cartes-mères intègrent une partie de ces fonctionnalités, il n'est donc pas toujours nécessaire de recourir à des cartes d'extension.

EN PRATIQUE **Vérifier que le matériel fonctionne**

Il n'est pas toujours évident de vérifier que le matériel fonctionne. En revanche, il est parfois simple de constater qu'il ne marche plus !
Un disque dur est constitué de plateaux rotatifs et de têtes de lectures qui se déplacent. Lorsque le disque dur est mis sous tension, il fait un bruit caractéristique dû à la rotation des plateaux. De plus, l'énergie dissipée entraîne un réchauffement du disque. Un disque alimenté qui reste froid et silencieux est vraisemblablement hors d'usage.
Les cartes réseau disposent souvent de LED qui indiquent l'état de la connexion. Si un câble est branché et qu'il aboutit sur un concentrateur (*hub*) ou un commutateur (*switch*) sous tension, une LED au moins sera allumée. Si aucune LED n'est allumée, soit la carte est défectueuse, soit le périphérique connecté à l'autre bout du câble est défectueux, soit le câble est défectueux. Il ne reste plus qu'à tester individuellement les composants incriminés.
Certaines cartes électroniques filles — les cartes vidéo 3D notamment — disposent de mécanismes de refroidissement intégré, souvent des radiateurs et des ventilateurs. Si le ventilateur ne tourne pas alors que la carte est sous tension, il est probable que la carte ait surchauffé et soit abîmée. Il en va de même pour le (ou les) processeur(s) situé(s) sur la carte mère.

Le démarreur : le BIOS

Le matériel seul n'est cependant pas autonome ; il est même totalement inutile sans qu'une partie logicielle permette d'en tirer parti. C'est le but du système d'exploitation et des applications — qui, de manière similaire, ne peuvent fonctionner sans un ordinateur pour les exécuter.

Il est donc nécessaire d'ajouter un élément de liaison, qui mette en relation le matériel et les logiciels, ne serait-ce qu'au démarrage de l'ordinateur. C'est le rôle principal du BIOS, qui est un petit logiciel intégré à la carte-mère de l'ordinateur et exécuté automatiquement à l'allumage. Sa tâche primordiale consiste à déterminer à quel logiciel passer la main. Il s'agit en général de trouver le premier disque dur contenant un secteur d'amorçage (souvent appelé MBR pour *Master Boot Record*), de charger ce secteur d'amorçage, et de l'exécuter. À partir de ce moment, le BIOS n'est généralement plus utilisé (jusqu'au démarrage suivant).

Le secteur d'amorçage contient à son tour un petit logiciel, le chargeur de démarrage, dont la tâche sera de trouver un système d'exploitation et de l'exécuter. Comme ce chargeur de démarrage n'est pas embarqué dans la carte-mère mais chargé depuis un disque dur (ou autre), il dispose de

OUTIL

Setup, l'outil de configuration du BIOS

Le BIOS contient également un logiciel appelé Setup, qui permet de configurer certains aspects de l'ordinateur. On pourra notamment choisir le périphérique de démarrage à favoriser (par exemple, le lecteur de disquettes ou de CD-Rom), régler l'horloge interne, etc. Pour lancer cet outil, il faut généralement appuyer sur une touche très tôt après la mise sous tension de l'ordinateur. C'est souvent *Suppr* ou *Échap*, plus rarement *F2* ou *F10*, mais la plupart du temps elle est indiquée à l'écran.

plus de possibilités que le chargeur du BIOS (ce qui explique pourquoi le BIOS ne charge pas le système d'exploitation directement). Le chargeur de démarrage (souvent Lilo ou GRUB sur les systèmes Linux) peut ainsi proposer de choisir quel système démarrer si plusieurs sont présents, avec un choix par défaut faute de réponse dans un délai imparti, avec des paramètres divers éventuellement saisis par l'utilisateur, etc. Il finit donc par trouver un noyau à démarrer, le charge en mémoire et l'exécute.

Le BIOS est également responsable de l'initialisation et de la détection d'un certain nombre de périphériques. Il détecte bien entendu les périphériques IDE/SATA (disques durs et lecteurs de CD-Roms/DVD-Roms) mais souvent aussi les périphériques PCI. Les périphériques détectés sont généralement listés de manière furtive au démarrage (l'appui sur la touche *Pause* permet souvent de figer l'écran pour l'analyser plus longuement). Si un des périphériques PCI installés n'y apparaît pas, c'est mauvais signe. Au pire le périphérique est défectueux, au mieux il fonctionne mais il est incompatible avec cette version du BIOS ou de la carte mère. Les spécifications PCI ont en effet évolué au fil du temps et il n'est pas impossible qu'une ancienne carte mère ne supporte pas une carte PCI récente.

Le noyau

Nous arrivons alors au premier logiciel qui va s'exécuter de manière durable (le BIOS et le chargeur de démarrage ne fonctionnent que quelques secondes chacun) : le noyau du système d'exploitation. Celui-ci prend alors le rôle de chef d'orchestre, pour assurer la coordination entre le matériel et les logiciels. Ce rôle inclut différentes tâches, notamment le pilotage du matériel, la gestion des processus, des utilisateurs et des permissions, le système de fichiers, etc. Il fournit ainsi une base commune aux programmes du système.

L'espace utilisateur

Bien que tout ce qui se passe au-dessus du noyau soit regroupé sous le vocable d'espace utilisateur, on peut encore différencier des couches logicielles ; mais leurs interactions étant plus complexes que précédemment, la différenciation n'est plus aussi simple. Un programme peut en effet faire appel à des bibliothèques qui font à leur tour appel au noyau, mais le flux des communications peut aussi mettre en jeu d'autres programmes, voire de multiples bibliothèques s'appelant l'une l'autre.

Quelques fonctions remplies par le noyau

Pilotage du matériel

Le noyau sert d'abord à contrôler les différents composants matériels, les recenser, les mettre en marche lors de l'initialisation de l'ordinateur, etc. Il les rend également disponibles pour les applications de plus haut niveau, avec une interface de programmation simplifiée : les logiciels peuvent ainsi utiliser les périphériques sans se préoccuper de détails de très bas niveau comme l'emplacement dans lequel est enfichée la carte-fille. L'interface de programmation offre également une couche d'abstraction qui permet par exemple à un logiciel de visiophonie de tirer parti d'une webcam de la même manière quels que soient sa marque et son modèle ; ce logiciel utilise simplement l'interface de programmation V4L (*Video for Linux*, le quatre se prononçant comme *for* en anglais), et c'est le noyau qui traduira les appels de fonction de cette interface en commandes spécifiques au type de webcam réellement utilisé.

Le noyau exporte de nombreuses informations sur le matériel qu'il a détecté par l'intermédiaire des systèmes de fichiers virtuels `/proc/` et `/sys/`. Plusieurs utilitaires permettent de synthétiser certaines de ces informations : citons **`lspci`** (du paquet `pciutils`) qui affiche la liste des périphériques PCI connectés, **`lsusb`** (du paquet `usbutils`) qui fait de même avec les périphériques USB et **`lspcmcia`** (du paquet `pcmciautils`) pour les cartes PCMCIA. Ces programmes sont très utiles quand il faut pouvoir identifier de manière certaine le modèle d'un périphérique. En outre cette identification unique permet de mieux cibler les recherches sur Internet et de trouver plus facilement des documents pertinents.

EXEMPLE Exemple d'informations fournies par lspci et lsusb

```
$ lspci
[...]
00:02.1 Display controller: Intel Corporation Mobile 915GM/GMS/910GML Express Graphics Controller (rev 03)
00:1c.0 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 1 (rev 03)
00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #1 (rev 03)
[...]
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5751 Gigabit Ethernet PCI Express (rev 01)
02:03.0 Network controller: Intel Corporation PRO/Wireless 2200BG Network Connection (rev 05)
$ lsusb
Bus 005 Device 004: ID 413c:a005 Dell Computer Corp.
Bus 005 Device 008: ID 413c:9001 Dell Computer Corp.
Bus 005 Device 007: ID 045e:00dd Microsoft Corp.
Bus 005 Device 006: ID 046d:c03d Logitech, Inc.
[...]
Bus 002 Device 004: ID 413c:8103 Dell Computer Corp. Wireless 350 Bluetooth
```

Les options `-v` de ces programmes permettent d'obtenir des informations beaucoup plus détaillées qui ne seront généralement pas nécessaires. Enfin, la commande **`lsdev`** (du paquet `procinfo`) permet de lister les différentes ressources de communication exploitées par les périphériques présents.

Bien souvent les applications accèdent aux périphériques par le biais de fichiers spéciaux qui sont créés dans `/dev/` (voir encadré « Droits d'accès à un périphérique » page 138). Il existe des fichiers spéciaux qui représentent les disques (par exemple `/dev/hda` et `/dev/sdc`), les partitions (`/dev/hda1` ou `/dev/sdc3`), la souris (`/dev/input/mouse0`), le clavier (`/dev/input/event0`), la carte son (`/dev/snd/*`), les ports série (`/dev/ttyS*`), etc.

Systèmes de fichiers

Un des aspects les plus visibles du noyau est celui des systèmes de fichiers. Les systèmes Unix intègrent en effet les différentes méthodes de stockage de fichiers dans une arborescence unique, ce qui permet aux utilisateurs (et aux applications) de stocker ou retrouver des données simplement grâce à leur emplacement dans cette arborescence.

Le point de départ de cette arborescence est la racine, `/`. Il s'agit d'un répertoire pouvant contenir des sous-répertoires, chacun étant identifié par son nom. Par exemple, le sous-répertoire `home` de `/` est noté `/home/` ; ce sous-répertoire peut à son tour contenir d'autres sous-répertoires, et ainsi de suite. Chaque répertoire peut également contenir des fichiers, qui contiendront les données réellement stockées. Le nom de fichier `/home/rmas/Desktop/hello.txt` désigne ainsi un fichier appelé `hello.txt`, stocké dans le sous-répertoire `Desktop` du sous-répertoire `rmas` du répertoire `home` présent à la racine. Le noyau fait alors la traduction entre ce système de nommage de fichiers et leur format de stockage physique sur disque.

Contrairement à d'autres systèmes, cette arborescence est unique et peut intégrer les données de plusieurs disques. L'un de ces disques est alors utilisé comme racine, les autres étant « montés » dans des répertoires de l'arborescence (la commande Unix qui permet cela est **`mount`**) ; ces autres disques sont alors accessibles sous ces « points de montage ». On peut ainsi déporter sur un deuxième disque dur les données personnelles des utilisateurs (qui sont traditionnellement stockés dans `/home/`). Ce disque contiendra alors les répertoires `rhertzog` et `rmas`. Une fois le disque monté dans `/home/`, ces répertoires deviendront accessibles aux emplacements habituels, et on pourra retrouver `/home/rmas/Desktop/hello.txt`.

Il existe différents systèmes de fichiers, qui correspondent à différentes manières de stocker physiquement les données sur les disques. Les plus connus sont *ext2*, *ext3* et *reiserfs*, mais il en existe d'autres. Par exemple, *vfat* est le système historiquement utilisé par les systèmes de type DOS

et Windows, et permet donc d'utiliser des disques durs sous Debian autant que sous Windows. Dans tous les cas, il faut préparer le système de fichiers avant de pouvoir le monter ; cette opération, fréquemment appelée formatage, est effectuée par le biais de commandes comme **`mkfs.ext3`** (**`mkfs`** étant une abréviation de *MaKe FileSystem*). Ces commandes prennent en paramètre le fichier de périphérique représentant la partition à formater (par exemple `/dev/hda1`). Cette opération destructrice n'est à exécuter qu'une seule fois sauf si l'on souhaite délibérément vider le contenu du système de fichiers et repartir de zéro.

Il existe même des systèmes de fichiers réseau, comme NFS, où les données ne sont pas stockées sur un disque local ; elles sont en effet transmises à un serveur sur le réseau, qui les stockera lui-même et les restituera à la demande ; l'abstraction du système de fichiers permet aux utilisateurs de ne pas avoir à s'en soucier : les fichiers resteront accessibles par leurs emplacements dans l'arborescence.

Fonctions partagées

Le noyau est également responsable de fonctions utilisées par tous les logiciels, et qu'il est judicieux de centraliser ainsi. Ces fonctions incluent notamment la gestion des systèmes de fichiers, qui permettent à une application d'ouvrir simplement un fichier en fonction de son nom, sans avoir à se préoccuper de l'emplacement physique du fichier (qui peut se trouver morcelé en plusieurs emplacements d'un disque dur, voire entre plusieurs disques durs, ou stocké à distance sur un serveur de fichiers) ; il s'agit également de fonctions de communication, que les applications pourront appeler pour échanger des informations à travers le réseau sans se soucier du mode de transport des données (qui pourront transiter sur un réseau local, ou une ligne téléphonique, ou un réseau sans fil, ou une combinaison de tout cela).

Gestion des processus

Un processus correspond à un programme en cours d'exécution. Ceci inclut une zone de mémoire dans laquelle est stocké le programme lui-même, mais également l'ensemble des données sur lesquelles le programme travaille. Le noyau est responsable de la création des processus et de leur suivi : lorsqu'un programme est lancé, le noyau met de côté cette zone de mémoire qu'il réserve au processus, y charge (depuis le disque) le code du programme et lance l'exécution. Il garde également des informations qui concernent ce processus, notamment un numéro d'identification (*pid*, pour *process identifier*).

NOTE

Systèmes multi-processeurs et assimilés

La restriction évoquée ci-contre est en réalité un cas particulier. La réelle restriction est qu'il ne peut s'exécuter à un instant donnée qu'un processus *par cœur de processeur*. Les systèmes multi-processeurs, multi-cœur ou proposant de l'*hyperthreading* permettent en effet à plusieurs processus d'être exécutés simultanément ; le même principe de découpage du temps en intervalles attribués à tour de rôle aux processus actifs reste appliqué, afin de pouvoir traiter le cas où le nombre de processus en cours est supérieur à celui des cœurs disponibles. Cette situation est loin d'être exceptionnelle : un système de base, même peu actif, a presque toujours quelques dizaines de processus en cours d'exécution.

Les noyaux de type Unix (dont fait partie Linux), comme la plupart des systèmes d'exploitation modernes, sont dits « multi-tâches », c'est-à-dire qu'ils permettent l'exécution « simultanée » de nombreux processus. En réalité, un seul processus peut fonctionner à un instant donné ; le noyau découpe alors le fil du temps en fines tranches et exécute les différents processus à tour de rôle. Comme ces intervalles de temps ont des durées très courtes (de l'ordre de la milliseconde), l'utilisateur a l'illusion de programmes s'exécutant en parallèle, alors qu'ils ne sont en réalité actifs que pendant certains intervalles, et suspendus le reste du temps. La tâche du noyau est d'ajuster ses mécanismes d'ordonnancement pour parfaire cette illusion tout en maximisant les performances globales du système : si les intervalles sont trop longs, l'application manquera de réactivité vis-à-vis de l'utilisateur ; s'ils sont trop courts, le système perdra du temps à basculer d'une tâche à l'autre trop souvent. Ces décisions peuvent être influencées par des notions de priorités affectées à un processus ; un processus de haute priorité bénéficiera pour s'exécuter d'intervalles de temps plus longs et plus fréquents qu'un processus de basse priorité.

Bien entendu, le noyau permet d'exécuter en parallèle plusieurs processus correspondant au même programme : chacun dispose alors de ses propres intervalles de temps pour s'exécuter, ainsi que de sa zone de mémoire réservée. Comme un processus n'a accès qu'à sa propre zone de mémoire, les données de chacun restent indépendantes.

Gestion des permissions

Les systèmes de type Unix sont également multi-utilisateurs. Ils intègrent donc une notion de droits permettant de séparer les utilisateurs entre eux, et d'autoriser ou non certaines actions en fonction de l'ensemble de droits dont on dispose. Le noyau gère donc, pour chaque processus, un ensemble de données permettant de vérifier les permissions de ce processus. En règle générale, il s'agit de « l'identité » sous laquelle tourne le processus, qui correspond le plus souvent au compte utilisateur qui a déclenché son exécution. Toute une série d'actions sont sujettes à l'approbation du noyau, et ne pourront être menées à bien par le processus que s'il dispose des permissions requises. Par exemple, l'opération d'ouverture d'un fichier est subordonnée à une vérification de la compatibilité entre les permissions du fichier et l'identité du processus (cet exemple particulier est détaillé en page 168).

L'espace utilisateur

On appelle espace utilisateur l'environnement d'exécution des processus normaux, par opposition aux processus qui font partie du noyau. Cela ne signifie pas pour autant que tous ces processus sont réellement lancés directement par l'utilisateur : un système normal exécute un certain nombre de « démons » avant même que l'utilisateur ouvre une session de travail.

Processus

Lorsque le noyau a terminé son initialisation, il lance le tout premier processus, **`init`**. Mais ce seul processus n'est généralement pas utile par lui-même. Les systèmes Unix fonctionnent donc avec tout un cycle de vie des processus.

Tout d'abord, un processus peut se dupliquer (on parle de *fork*). Le noyau alloue alors une nouvelle zone de mémoire pour le deuxième processus, de contenu identique à celle du premier, et se retrouve simplement avec un processus supplémentaire à gérer. À ce moment précis, la seule différence entre les deux processus est leur *pid*. Par convention, le nouveau processus est appelé le fils, alors que celui dont le *pid* n'a pas changé est appelé le père.

Il arrive que le processus fils reste tel quel et « vive sa vie », indépendamment de son père, avec ses propres données correspondant au programme initial. Mais le cas le plus fréquent est que ce fils exécute un autre programme ; à de rares exceptions près, sa zone mémoire est alors simplement remplacée par le nouveau programme, dont l'exécution démarre. C'est ainsi qu'une des premières actions du processus n°1 est de se dupliquer (il existe donc temporairement deux processus correspondant au programme **`init`**) ; le processus fils ainsi créé se remplace alors par le premier script d'initialisation du système, `/etc/init.d/rcS`. Ce script va à son tour se dupliquer puis exécuter différents autres programmes, pour qu'arrive un moment dans la filiation où un processus déclenchera une interface graphique pour l'utilisateur (la séquence des événements est décrite avec plus de détails au chapitre 9, en page 160).

Lorsqu'un processus finit la tâche qui lui était dévolue, il se termine. Le noyau récupère alors la mémoire qui était affectée à ce processus, et cesse de lui distribuer des intervalles de temps d'exécution. Le processus père est informé de la destruction du fils : cela permet entre autres au père d'attendre la complétion d'une tâche sous-traitée. On retrouve ce mode de fonctionnement dans les interpréteurs de commandes (shells) : lorsque l'on tape une commande dans un shell, on ne retrouve l'invite que lorsqu'elle s'est terminée. La plupart des shells permettent cepen-

dant de ne pas attendre la fin de l'exécution d'une commande : il suffit pour cela de faire suivre le nom du programme à exécuter par &. On retrouve alors l'invite aussitôt, ce qui peut poser des problèmes si la commande a des données à afficher.

Démons

Un démon est un processus lancé automatiquement au démarrage et qui fonctionne en tâche de fond pour accomplir certaines tâches de maintenance ou fournir des services aux autres processus. Cette notion de « tâche de fond » est arbitraire, et ne correspond à rien de particulier du point de vue du système : ce sont des processus comme les autres, qui sont exécutés chacun à leur tour pendant un bref intervalle de temps de la même manière que les applications visibles. La distinction est simplement humaine : un processus qui fonctionne sans interaction avec l'utilisateur (sans interface graphique, notamment) est dit fonctionner en tâche de fond ou en tant que démon.

Plusieurs de ces démons sont détaillés dans le chapitre 9.

VOCABULAIRE Démon, un terme péjoratif ?

Le terme démon est en réalité une transcription un peu hâtive de l'anglais *daemon*. Bien que l'origine grecque de ce mot ait également donné le mot *demon*, au sens de créature diabolique, le *daemon* est simplement à interpréter comme un aide, un auxiliaire (tout en gardant une dimension surnaturelle). Il n'y a pas en français de mot réellement adapté à ce concept, le sens du *daemon* anglais s'est donc retrouvé projeté sur le « démon » français, et l'usage a consacré ce choix bien qu'il ne soit pas très heureux.

Communications entre processus

Qu'il s'agisse de démons ou d'applications interactives, un processus isolé n'est souvent pas très utile. Il existe donc différentes méthodes permettant à des processus séparés de communiquer entre eux, soit pour s'échanger des données soit pour se contrôler l'un l'autre. Le terme générique les désignant est *InterProcess Communications* (IPC) soit « communications inter-processus ».

Le système le plus simple est le fichier : le processus qui souhaite émettre des données les écrit dans un fichier dont le nom est convenu à l'avance ; le processus destinataire n'a alors qu'à lire ce fichier pour y récupérer les données.

Pour éviter que les données soient stockées sur un disque dur, on peut également utiliser un tuyau ou tube (*pipe* en anglais). Il s'agit simplement d'un système de communications où des octets écrits à un bout ressortent tels quels à l'autre bout. Si les deux extrémités sont contrôlées par deux processus différents, on obtient un canal de communication simple et pratique. Les tubes se décomposent en deux catégories. Un tube nommé dispose d'une entrée spéciale dans le système de fichiers (bien que les données qui y transitent n'y soient pas stockées), et les deux processus peuvent donc l'ouvrir indépendamment l'un de l'autre, si l'emplacement du tube nommé est connu. Dans les cas où l'on cherche à faire communiquer deux processus apparentés (par exemple un père et son

fils), il est possible au père de créer un tube anonyme, dont héritera son fils après le *fork* ; les deux processus pourront alors s'échanger des données sans passer par le système de fichiers.

EN PRATIQUE **Un exemple concret**

Étudions ce qui se passe lorsqu'on lance une commande complexe (un *pipeline*) dans un shell. Supposons que nous ayons un processus **bash** (le shell standard sous Debian), de *pid* 4374, dans lequel nous tapons la commande `ls | sort`.
Le shell commence par interpréter la commande saisie. En l'occurrence, il s'agit de deux programmes (`ls` et `sort`), avec un flux de données de l'un vers l'autre (noté par le caractère |, dit *pipe*). **bash** crée donc un tube anonyme (qui n'existe pour l'instant que pour lui seul).
Puis il se duplique ; on obtient donc un nouveau processus **bash**, de *pid* 4521 (les *pids* sont de simples numéros abstraits, et n'ont généralement pas de signification particulière). Ce processus n°4521 hérite du tuyau anonyme, il pourra donc écrire du côté « entrée » ; **bash** redirige d'ailleurs le flux de sortie standard vers cette entrée du tuyau. Il se remplace ensuite par le programme `ls`, qui va lister le contenu du répertoire courant ; comme il écrit sur sa sortie standard et que celle-ci a été au préalable redirigée, le résultat est effectivement envoyé dans le tuyau.
Une opération similaire est effectuée pour la deuxième commande : **bash** se duplique de nouveau, on obtient alors un nouveau processus **bash** de numéro 4522. Comme ce dernier est également un fils du n°4374, il hérite aussi du tuyau ; **bash** branche alors la sortie du tuyau sur son flux d'entrée standard, puis se remplace par le programme `sort`, dont la vocation est de trier les données reçues et d'afficher le résultat.
Toutes les pièces sont maintenant en place : `ls` envoie la liste des fichiers du répertoire courant dans le tuyau ; `sort` lit cette liste, puis la trie par ordre alphabétique, et affiche le résultat. Les processus n°4521 et n°4522 se terminent alors, et le 4374, qui s'était mis en attente, reprend la main et affiche l'invite pour permettre à l'utilisateur de saisir une nouvelle commande.

Mais toutes les communications inter-processus ne servent pas à faire transiter des flux de données. Il arrive également que des applications aient simplement besoin de se transmettre des messages comme « suspendre l'exécution » ou « reprendre ». Unix (et donc Linux) fournit pour cela un mécanisme de signaux, par lequel un processus peut simplement envoyer un signal prédéfini (parmi une liste fixe de quelques dizaines de signaux) à un autre, simplement en connaissant son *pid*.

Pour des communications plus complexes, il existe aussi des mécanismes par lesquels un processus peut par exemple ouvrir l'accès d'une partie de sa zone mémoire à d'autres ; cette mémoire est alors partagée entre plusieurs processus, ce qui permet de faire passer des données de l'un à l'autre.

Enfin, les connexions par le réseau peuvent également servir à faire communiquer différents processus ; ils peuvent même, dans ce cas, s'exécuter sur des ordinateurs différents (voire séparés de milliers de kilomètres).

Tous ces mécanismes sont utilisés, à des degrés divers, dans le fonctionnement normal d'un système Unix typique.

VOCABULAIRE **Bibliothèque ou librairie ?**

Ici encore, l'anglais nous joue des tours et déborde un peu sur le français. Le mot « bibliothèque » se traduit en anglais par *library*, un faux ami notoire ; il arrive donc, malgré les récriminations des puristes, que l'on entende parler de « librairie de fonctions ».

CULTURE

La méthode Unix : une chose à la fois

Un des concepts qui sous-tend le fonctionnement général des systèmes d'exploitation de la famille Unix est que chaque outil ne devrait faire qu'une chose, mais la faire bien, les applications pouvant alors réutiliser ces outils et construire une logique plus poussée par-dessus. Cela transparaît dans de nombreux domaines. Les scripts shell sont peut-être le meilleur exemple, qui assemblent en des séquences complexes des outils très simples (**`grep`**, **`wc`**, **`sort`**, **`uniq`**, etc.). Une autre mise en pratique de cette philosophie est visible dans les bibliothèques de code : la *libpng* permet de lire et d'écrire des images au format PNG, avec différentes options et de différentes manières, mais elle ne fait que cela ; pas question pour elle de proposer des fonctions d'affichage ou d'édition.

Bibliothèques

Les bibliothèques de fonctions jouent un rôle crucial dans le fonctionnement d'un système d'exploitation Unix. Ce ne sont pas à proprement parler des programmes, puisqu'elles ne s'exécutent pas indépendamment, mais des collections de fragments de programmes qui peuvent être utilisés par des programmes classiques. Parmi les bibliothèques les plus courantes, citons par exemple :

- la bibliothèque C standard (*glibc*), qui contient des fonctions de base telles celles permettant d'ouvrir des fichiers ou des connexions réseau, mais aussi de faciliter les interactions avec le noyau ;
- les boîtes à outils graphiques (*toolkits*), Gtk+ et Qt, qui permettent à de nombreux programmes de réutiliser les objets graphiques qu'elles proposent ;
- la bibliothèque *libpng*, qui permet de charger, d'interpréter et sauvegarder des images au format PNG.

L'existence de ces bibliothèques permet aux applications de réutiliser du code existant ; leur développement en est simplifié d'autant, surtout lorsque de nombreuses applications font appel aux mêmes fonctions. Comme les bibliothèques sont souvent développées par des personnes différentes, le développement global du système est ainsi plus proche de la philosophie historique d'Unix.

De plus, ces bibliothèques sont souvent dites « partagées », parce que le noyau est capable de ne les charger qu'une fois en mémoire même si plusieurs processus y font appel. Si le code qu'elles contiennent était au contraire intégré dans les applications, il serait présent en mémoire autant de fois qu'il y a de processus qui l'utilisent.