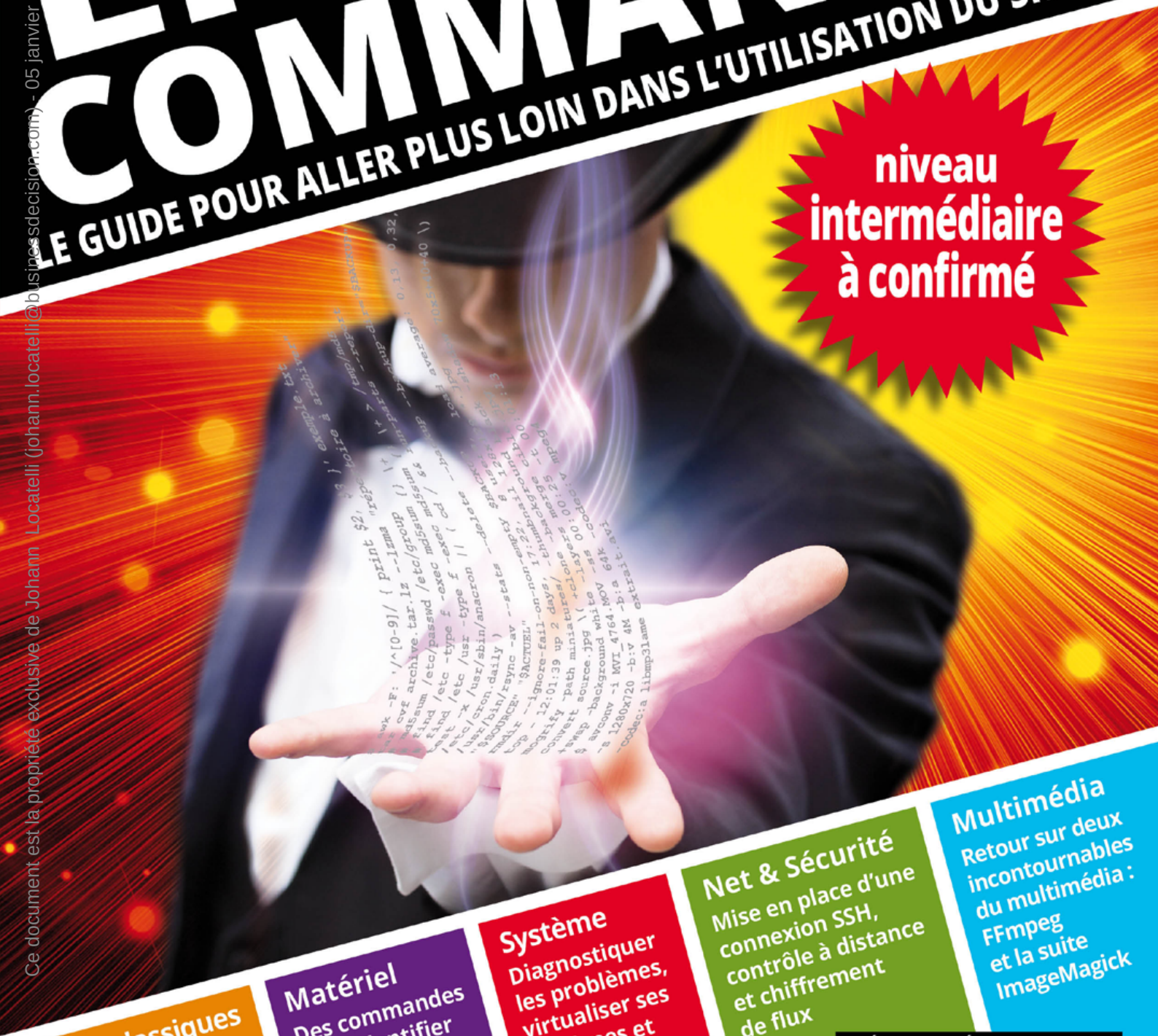
LES GUIDES DE
GNU LINUX MAGAZINE / FRANCE
HORS-SÉRIE N°72

France METRO : 12,90 € — CH : 18,00 CHF — BEL/PORT.CONT : 13,90 € — DOM TOM : 13,90 € — CAN : 18,00 $ cad — MAR : 130 MAD

LIGNE DE COMMANDES

LE GUIDE POUR ALLER PLUS LOIN DANS L'UTILISATION DU SHELL !

niveau intermédiaire à confirmé



Les classiques
Les outils indispensables pour manipuler et sécuriser ses données

Matériel
Des commandes pour identifier précisément et exploiter son matériel

Système
Diagnostiquer les problèmes, virtualiser ses systèmes et opter pour la flexibilité

Net & Sécurité
Mise en place d'une connexion SSH, contrôle à distance et chiffrement de flux

Multimédia
Retour sur deux incontournables du multimédia : FFmpeg et la suite ImageMagick

Édité par Les Éditions Diamond
L 15066 - 72 H - F: 12,90 € - RD
boutique.ed-diamond.com

Retrouvez toutes nos publications

LES ÉDITIONS DIAMOND

sur **boutique.ed-diamond.com**

GNU/Linux Magazine Hors-Série
est édité par **Les Éditions Diamond**

B.P. 20142 / 67603 Sélestat Cedex

Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21

E-mail : cial@ed-diamond.com
lecteurs@gnulinuxmag.com

Service commercial : abo@gnulinuxmag.com

Sites : www.gnulinuxmag.com
boutique.ed-diamond.com

Directeur de publication : Arnaud Metzler

Chef des rédactions : Denis Bodor

Rédacteur en chef : Tristan Colombo

Remerciements à Sébastien Maccagnoni-Munch

Conception graphique : Kathrin Scali & Jéremy Gall

Responsable publicité : Tél. : 03 67 10 00 27

Service abonnement : Tél. : 03 67 10 00 20

Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias : Tél. : 05 34 52 34 01

IMPRIMÉ en Allemagne - PRINTED in Germany

Dépôt légal : A parution

N° ISSN : 0183-0864

Commission Paritaire : K78 976

Périodicité : Bimestrielle

Prix de vente : 12,90 Euros

Les articles non signés contenus dans ce numéro ont été rédigés par les membres de l'équipe rédactionnelle des Éditions Diamond.

PRÉFACE

De nos jours, l'écrasante majorité des systèmes informatiques proposent une interface graphique afin de simplifier, en apparence, le fonctionnement des ordinateurs. Mais les interfaces graphiques ont un défaut très pesant : elles ne permettent de faire que ce pour quoi elles sont prévues, car elles ne sont pas interconnectables.

La ligne de commandes, c'est « l'ancêtre » des interfaces informatiques : on tape des instructions au clavier, celles-ci sont exécutées et leur réponse est affichée. Mais des couches intermédiaires rendent ce fonctionnement réellement puissant. En réalité, un logiciel n'attend pas des entrées « du clavier » : il attend des données « sur l'entrée standard » ; de même, il n'affiche pas « à l'écran » ; il retourne des données « sur la sortie standard ». Le fait est que, dans le cadre d'un terminal, le clavier est dirigé vers l'entrée standard et la sortie standard est renvoyée vers l'écran. Notons aussi que toutes ces communications ne sont que des échanges de texte : il n'y a pas de format particulier à respecter, une entrée et une sortie ce n'est que du flux de texte.

Grâce aux opérateurs de redirection, on peut alors envoyer (rediriger) la sortie (textuelle) d'un programme sur l'entrée (textuelle) d'un autre. On parle alors de « KISS », pour « Keep It Simple, Stupid » : chaque commande ne sait faire qu'une chose, mais elle la fait bien. C'est l'enchaînement de ces commandes qui fait la puissance de la ligne de commandes. C'est par exemple tout l'intérêt de commandes comme **`sed`** et **`awk`** : elles traitent et transforment du texte.

En ligne de commandes, on peut aussi simplement vouloir exécuter des actions sans devoir lancer une interface graphique, à distance par exemple : il est largement plus facile de faire transiter du « texte pur » qu'une interface graphique, qui n'est après tout pas très différente d'une vidéo : des mises à jour plusieurs fois par seconde, des données lourdes à transférer, tout cela nécessitant une bande passante imposante. On se rend compte alors qu'on peut tout faire en ligne de commandes, l'interface graphique n'étant qu'une surcouche rendant certaines tâches plus agréables.

Pour toutes ces raisons (et bien d'autres encore), la ligne de commandes a de beaux jours devant elle : simple mais puissante, elle est bien plus flexible qu'une interface graphique figée, qui n'est capable que de faire ce pour quoi elle est prévue...

La Rédaction

Sommaire

GNU/Linux Magazine
Hors-Série
N°72

niveau intermédiaire à confirmé

LIGNE DE COMMANDES

1

LES CLASSIQUES

2

MATÉRIEL

3 SYSTÈME

4 NET & SÉCURITÉ

5 MULTIMÉDIA

1

1 LES CLASSIQUES

À découvrir dans cette partie...

1.1 Traitement de données avec sed et awk

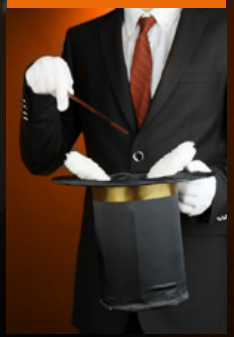
sed et awk sont deux commandes communément utilisées pour traiter des données sous forme textuelle : découpages, remplacements, identification de textes sont leur spécialité. p. 08

1.2 Automatiser des tâches récurrentes avec cron

Un bon informaticien est un informaticien fainéant. Il automatisera alors de nombreuses manipulations, grâce à l'outil système cron, dont c'est le rôle. Apprenons à l'utiliser. p. 18

1.3 Synchroniser ses données avec rsync

Lorsque l'on a plusieurs ordinateurs, on a envie d'avoir accès aux mêmes données. Plusieurs cas de figure peuvent se présenter, rsync est une solution de synchronisation de données parmi d'autres. Ses fonctionnalités permettent aussi de l'utiliser comme base d'une politique de sauvegarde. p. 24



1.4 Surveiller les modifications au sein des fichiers

Sur le disque, un système d'exploitation ce n'est, après tout, qu'un ensemble de fichiers. Quelle meilleure façon de s'assurer que le système n'est pas corrompu que de surveiller les fichiers qui le composent ? On abordera ici une méthode de surveillance de fichiers, afin de s'assurer qu'ils n'ont pas été modifiés par autrui. p. 30

1 LES CLASSIQUES

TRAITEMENT DE DONNÉES AVEC SED ET AWK

Dans le cadre d'un traitement automatisé (ou semi-automatisé) de données, on peut souvent être amené à utiliser les commandes sed et awk. Celles-ci permettent de modifier des flux de données (la sortie d'une quelconque autre commande) grâce à diverses instructions. On peut alors simplifier le traitement de tâches répétitives.

sed et **awk** sont deux commandes qui existent depuis des dizaines d'années. Plus précisément, elles sont apparues respectivement en 1973 et en 1977. Ces outils sont encore largement utilisés aujourd'hui, car ils répondent toujours très bien aux besoins de l'informatique : traiter et transformer des données.

Ils ont comme objectif la modification de flux de données textuelles (on inclut les fichiers texte dans ce concept), permettant d'automatiser des tâches que l'on ferait interactivement dans un éditeur de texte. Comme la majorité des outils en ligne de commandes, ils prennent les données à traiter sur leur entrée standard et retournent le résultat sur leur sortie standard.

Nous aborderons dans cet article de nombreux exemples, ceux-ci se basent tous sur un fichier **exemple.txt** de 8 lignes dont le contenu est le suivant :

Fichier

```
Ceci est un fichier d'exemple.
Une ligne commençant par une majuscule et terminant par un point.
une ligne finissant par un point.
Une ligne commençant par une majuscule
Une liste de valeurs séparées par un double-point
12:1234:25:36.5:22
135:1.5:22:24:18:111
5.1:2.5:13:12:11
```

À savoir

sed signifie simplement « stream editor », ou « éditeur de flux » en français. Le nom awk, quant à lui, est tiré des noms de ses co-créateurs : Alfred Aho, Peter Weinberger et Brian Kernighan.

1. SED, L'ÉDITEUR DE FLUX

Comme indiqué, **sed** attend les données à traiter sur son entrée standard. Il est également capable de traiter directement un ou plusieurs fichier(s) texte, il suffit alors de donner le(s) nom(s) de fichier(s) comme dernier(s) argument(s) à la commande.

sed traite les flux ligne par ligne : ça le rend notamment très performant sur de gros volumes de données, car il n'a pas besoin de charger l'ensemble d'un fichier en mémoire avant de le traiter.

1.1 Fonctionnement

Lorsque l'on exécute **sed**, on lui spécifie une ou plusieurs commande(s) sur la base desquelles il doit traiter les données qu'il reçoit en entrée.

Typiquement, on exécutera **sed** de la manière suivante pour travailler sur un fichier :

Terminal
```
sed <options> <motifs et commandes> <fichier>
```

ou, pour travailler sur un flux sur l'entrée standard :

Terminal
```
[...] | sed <options> <motifs et commandes>
```

Lorsque l'on donne plusieurs commandes à **sed**, on les sépare par un point-virgule (**;**) ; nous en verrons des exemples plus loin.

1.2 Options

sed accepte différentes options qui modifient son comportement, dont voici les plus importantes :

- l'option **-n** permet de ne pas afficher le résultat des traitements, sauf si la commande **p** est utilisée simultanément (voir plus bas) ;
- l'option **-f** permet de fournir un fichier de script au lieu de donner les commandes **sed** en argument (voir plus loin) ;
- l'option **-i** permet de modifier un fichier « en place » (modifie directement le fichier lui-même, **sed** ne retourne alors rien sur la sortie standard).

1.3 Motifs

Les motifs permettent d'indiquer à **sed** qu'il ne doit travailler que sur certaines lignes des données entrantes. Les commandes ne seront alors appliquées qu'aux lignes qui correspondent au motif. Ce motif peut revêtir plusieurs formes :

- Tout d'abord, il est optionnel : si on ne précise pas de motif, **sed** « travaille » sur toutes les lignes du flux ;
- Si le motif est un simple numéro (ou le caractère **$**), alors **sed** traite uniquement la ligne concernée (le caractère **$** désignant la dernière ligne du fichier) ;
- Si le motif est composé de deux numéros séparés par une virgule (**n1,n2**), alors **sed** agit sur l'ensemble des lignes comprises entre celles-ci (de la ligne **n1** à la ligne **n2**) ;
- Si le motif est une expression rationnelle (regexp), alors **sed** ne modifie que les lignes correspondant à l'expression (une expression rationnelle est encadrée de slashes **/**) ;
- Si le motif est un ensemble de deux expressions rationnelles séparées par une virgule, alors **sed** adopte le même comportement qu'avec des nombres et l'ensemble des lignes entre celles qui correspondent sont traitées.

Voici quelques exemples de motifs, appliqués au fichier **exemple.txt** :

À retenir

Attention, lorsque l'on utilise l'option -i de sed, on écrase le fichier d'origine. Par conséquent, si les arguments de sed sont mal formés, on peut perdre des données. C'est pourquoi on peut ajouter un suffixe après -i, de cette manière sed fera une sauvegarde de la version précédente du fichier, par exemple avec l'option -i.bak (le fichier exemple.txt serait copié dans exemple.txt.bak avant traitement).

Motif	Lignes correspondantes
`5,7`	Une liste de valeurs séparées par un double-point 12:1234:25:36.5:22 135:1.5:22:24:18:111
`6,$`	12:1234:25:36.5:22 135:1.5:22:24:18:111 5.1:2.5:13:12:11
`2`	Une ligne commençant par une majuscule et terminant par un point.
`/^[A-Z]/`	Ceci est un fichier d'exemple. Une ligne commençant par une majuscule et terminant par un point. Une ligne commençant par une majuscule Une liste de valeurs séparées par un double-point
`/^[a-z]/,/liste/`	une ligne finissant par un point. Une ligne commençant par une majuscule Une liste de valeurs séparées par un double-point

Une fois les lignes identifiées, il faut leur appliquer une action à effectuer ; cela se fait grâce aux commandes. Prenons par exemple la commande la plus simple : **p**.
Celle-ci a pour seul objectif d'afficher (*print*) la ligne correspondante.

Voici un exemple simple de commande **sed** :

Terminal

```
$ sed '/^Une/p' exemple.txt
Ceci est un fichier d'exemple.
Une ligne commençant par une majuscule et terminant par un point.
Une ligne commençant par une majuscule et terminant par un point.
une ligne finissant par un point.
Une ligne commençant par une majuscule
Une ligne commençant par une majuscule
Une liste de valeurs séparées par un double-point
Une liste de valeurs séparées par un double-point
12:1234:25:36.5:22
135:1.5:22:24:18:111
5.1:2.5:13:12:11
```

On constate que **sed** retourne toutes les lignes (car on n'utilise pas l'option **-n**), celles qui correspondent à l'expression rationnelle (**/^Une/** signifie « toute ligne qui commence par **Une** ») étant affichées une deuxième fois par le biais de la commande **p**.

La même commande en utilisant l'option **-n** donne :

Terminal

```
$ sed -n '/^Une/p' exemple.txt
Une ligne commençant par une majuscule et terminant par un point.
Une ligne commençant par une majuscule
Une liste de valeurs séparées par un double-point
```

1.4 Commandes

La vraie puissance de **sed** provient de ses commandes, qui lui permettent de manipuler les lignes de données de différentes manières. Nous verrons ici une partie de ces commandes, accompagnées d'exemples, mais de nombreuses autres existent.

Notons qu'on ne met aucun séparateur entre le motif et la commande : **sed** sait comment identifier l'un et l'autre grâce à sa syntaxe spécifique.

Commande	Effet	Exemple

`=`	Affiche le numéro de la ligne	`$ sed -n '/^Une/=' exemple.txt`
		`2`
		`4`
		`5`

`a<texte>`	Ajoute une ligne de texte `<texte>` après celle(s) qui correspond(ent) au motif	`$ sed '/^Ceci/aDeuxième ligne' exemple.txt`
		`Ceci est un fichier d'exemple.`
		`Deuxième ligne`
		`Une ligne commençant par une majuscule et terminant par un point.`
		`[...]`

`i<texte>`	Ajoute une ligne de texte `<texte>` avant celle(s) qui correspond(ent) au motif	`$ sed '/^12/iListe de nombres:' exemple.txt`
		`[...]`
		`Une liste de valeurs séparées par un double-point`
		`Liste de nombres:`
		`12:1234:25:36.5:22`
		`135:1.5:22:24:18:111`
		`5.1:2.5:13:12:11`

`c<texte>`	Remplace les lignes correspondant au motif par le texte donné	`$ sed '/Une/cIl était une ligne...' exemple.txt`
		`Ceci est un fichier d'exemple.`
		`Il était une ligne...`
		`une ligne finissant par un point.`
		`Il était une ligne...`
		`Il était une ligne...`
		`12:1234:25:36.5:22`
		`135:1.5:22:24:18:111`
		`5.1:2.5:13:12:11`

`d`	Supprime les lignes correspondant au motif	`$ sed '/Une/d' exemple.txt`
		`Ceci est un fichier d'exemple.`
		`une ligne finissant par un point.`
		`12:1234:25:36.5:22`
		`135:1.5:22:24:18:111`
		`5.1:2.5:13:12:11`

`s/<expression>/<remplacement>/`	Substitue la première portion de texte correspondant à l'expression rationnelle par le texte de remplacement.	`$ sed 's/^une/Il manquait une majuscule dans la/' exemple.txt`
		`Ceci est un fichier d'exemple.`
		`Une ligne commençant par une majuscule et terminant par un point.`
		`Il manquait une majuscule dans la ligne finissant par un point.`
		`Une ligne commençant par une majuscule`
		`[...]`

`s/<expression>/<remplacement>/g`	Substitue toute portion de texte correspondant à l'expression rationnelle par le texte de remplacement	`$ sed -n '/terminant/s/par/avec/p' exemple.txt`
		`Une ligne commençant avec une majuscule et terminant par un point.`
		`$ sed -n '/terminant/s/par/avec/gp' exemple.txt`
		`Une ligne commençant avec une majuscule et terminant avec un point.`

`y/<caractères>/<caractères>/`	Remplace tout caractère du premier groupe par le caractère correspondant dans le second groupe	`$ sed 'y/abeilost/48311057/' exemple.txt`
		`C3c1 357 un f1ch13r d'3x3mp13.`
		`Un3 11gn3 c0mm3nç4n7 p4r un3 m4ju5cu13 37`
		`73rm1n4n7 p4r un p01n7.`
		`un3 11gn3 f1n1554n7 p4r un p01n7.`
		`Un3 11gn3 c0mm3nç4n7 p4r un3 m4ju5cu13`
		`Un3 11573 d3 v413ur5 5ép4ré35 p4r un d0u813-p01n7`
		`[...]`

`!<commande>`	Le point d'exclamation permet d'utiliser la négation du motif avant d'exécuter la commande : celle-ci s'exécute donc sur les lignes **ne correspondant pas** au motif.	`$ sed -n '/ligne/!p' exemple.txt`
		`Ceci est un fichier d'exemple.`
		`Une liste de valeurs séparées par un double-point`
		`12:1234:25:36.5:22`
		`135:1.5:22:24:18:111`
		`5.1:2.5:13:12:11`

Notes :

- Pour les commandes **a**, **i** et **c**, on peut inclure des retours à la ligne en les faisant précéder par un anti-slash **** ;
- Dans les commandes **s** et **y**, on peut remplacer le séparateur **/** par n'importe quel autre caractère, **sed** utilise simplement le caractère immédiatement après le **s** ou le **y** comme séparateur ;
- Comme on l'a vu dans les exemples, le motif n'est pas indispensable : dans l'exemple de la commande **s**, on ne précise pas de motif, cela veut dire que la commande se fait sur toutes les lignes... Mais une seule d'entre elles correspond à l'expression rationnelle **^une**.

1.5 Scripts

On peut cumuler plusieurs commandes **sed** en une seule ligne, en les séparant par un point-virgule :

Terminal

```
$ sed '/Une/d;s/12/AB/g' exemple.txt
Ceci est un fichier d'exemple.
une ligne finissant par un point.
AB:AB34:25:36.5:22
135:1.5:22:24:18:111
5.1:2.5:13:AB:11
```

Ici, on efface les lignes contenant le mot « Une » et un remplace « 12 » par « AB » partout où cette suite de caractères est trouvée.

Mais on peut également fournir à **sed** un script placé dans un fichier, où chaque ligne peut contenir une commande pour **sed** ; ce fichier est alors fourni à **sed** grâce à l'option **-f**.

Créons par exemple un fichier **commandes.sed** :

Fichier

```
# Commandes pour sed
# Un dièse en début de ligne indique un commentaire

# Supprimer les lignes commençant par un chiffre
s/^[0-9]/d

# Remplacer " majuscule " par " MAJUSCULE "
s/majuscule/MAJUSCULE/

# Suppression des lignes ne se terminant pas par un point
/\.$/!d
```

On obtient alors le résultat suivant :

Terminal

```
$ sed -f commandes.sed exemple.txt
Ceci est un fichier d'exemple.
Une ligne commençant par une MAJUSCULE et terminant par un
point.
une ligne finissant par un point.
```

À savoir

La commande de substitution est la plus connue des commandes de sed. C'est généralement celle que l'on utilise le plus et parfois celle que l'on apprend en premier.

À retenir

Lorsque l'on se demande laquelle des deux commandes il faut utiliser pour traiter du texte, il y a deux paramètres à prendre en compte :

- **d'une part, quand le fichier ou le flux est essentiellement composé de données tabulaires, awk est plus adapté, car il est conçu pour traiter des données placées dans des champs, séparées par des séparateurs – sed a beaucoup moins de facilité à traiter des données de ce type ;**
- **d'autre part, quand les besoins en termes de traitement de données sont complexes (mémorisation de données pour les reprendre quelques lignes plus loin par exemple, ou utilisation de boucles), awk est à préférer, car avec sed, il faudrait passer par un script shell ;**
- **dans la majorité des cas peu complexes, sed reste plus facile à utiliser.**

2. AWK, LE LANGAGE DE MANIPULATION DE TEXTE

Bien que présentant le même objectif que **sed**, la commande **awk** est beaucoup plus puissante, car elle se base sur un « vrai » langage de programmation (celui-ci propose des structures conditionnelles, des boucles, du stockage de données dans des variables, etc.). Ce langage reste cependant très simple, sa syntaxe se rapprochant de celle de **sed** pour la recherche de motifs et de la syntaxe du C dont il est inspiré (cela ne veut pas dire qu'il faut savoir développer en C pour utiliser **awk**, et savoir utiliser **awk** ne fera pas de vous un développeur C !).

Il existe plusieurs versions de **awk** et, en général, sur les distributions Linux, on utilise **gawk**, la version de **awk** proposée dans le cadre du projet GNU ; c'est cette version que l'on va aborder.

Pour une simple recherche de motif, **awk** s'utilise comme **sed** et affiche les lignes correspondant au motif recherché :

Terminal
```
$ awk '/^une/' exemple.txt
une ligne finissant par un point.
```

2.1 Fonctionnement

La commande **awk** attend des informations de deux manières différentes :

- On peut donner les instructions (le programme) à exécuter directement dans la ligne de commandeS :

Terminal
```
awk <options> <programme> <fichier>
```

- On peut donner le nom d'un fichier dans lequel lire le programme :

Terminal
```
awk <options> -f <fichier programme> <fichier à traiter>
```

Comme avec **sed**, dans les deux cas on peut omettre le nom de fichier, auquel cas **awk** prend les données sur son entrée standard.

2.2 Options

Les options acceptées par **awk** sont nombreuses, on peut notamment retenir les suivantes :

- L'option **-F** permet de définir un séparateur de champ différent – en effet, **awk** est capable d'extraire très facilement des données tabulaires tant que celles-ci sont séparées par un séparateur bien identifié. Par défaut, **awk** considère que n'importe quel espacement (un ou plusieurs caractère(s) espace, tabulation...) est un séparateur ;
- L'option **-v** permet d'assigner une valeur à un nom de variable avant l'exécution du programme, cette valeur sera alors accessible au sein du programme **awk** ;

⇨ L'option **-N** permet de demander à **awk** de reconnaître le séparateur de décimal selon la locale (la langue) utilisée – c'est utile par exemple quand on traite un fichier composé de nombres décimaux écrits en français, avec une virgule au lieu d'un point.

2.3 Structure des programmes

Comme nous l'avons indiqué, **awk** utilise un langage de programmation, qui suit une syntaxe un peu particulière ; comme avec **sed**, **awk** exécute des instructions (composées de fonctions et séparées par un point-virgule) suivant la reconnaissance d'un motif dans une ligne ; la syntaxe en est la suivante :

Fichier
```
<motif> { <fonction>;<fonction>[...] }
```

On peut mettre les actions sur plusieurs lignes, par contre l'accolade ouvrante doit rester sur la même ligne que le motif :

Fichier
```
<motif> {
  <fonction>
  <fonction>
}
```

De plus, si on ne spécifie pas de motif, le bloc est exécuté pour chaque ligne rencontrée (suivant la même logique que la commande **sed**).

Par exemple, la plus simple des fonctions est **print**, qui affiche une chaîne de caractères :

Terminal
```
$ awk "/^Une/ { print \"J'ai trouvé une ligne commençant par " Une
".\" }" exemple.txt
J'ai trouvé une ligne commençant par " Une ".
J'ai trouvé une ligne commençant par " Une ".
J'ai trouvé une ligne commençant par " Une ".
```

Ce programme **awk** affiche cette phrase pour chaque ligne correspondant au motif **^Une** (celui-ci signifie « la ligne commence par «Une» », vous l'aurez compris).

Il existe deux blocs spéciaux, où le motif est remplacé par le simple mot **BEGIN** ou **END** :

⇨ **BEGIN** définit un bloc à exécuter avant d'analyser le flux ou le fichier ;

⇨ **END** définit un bloc à exécuter après l'analyse.

Si on utilise par exemple le fichier **commandes.awk** suivant :

Fichier
```
BEGIN { print "Lecture d'un fichier avec awk" }
/^une/ {
  print "J'ai trouvé une ligne qui commence par " une "."
}
/^[0-9]/ { print "J'ai trouvé une ligne qui commence par un chiffre." }
END { print "Fin de la lecture du fichier !" }
```

On obtient le résultat suivant :

Terminal
```
$ awk -f commandes.awk exemple.txt
Lecture d'un fichier avec awk
J'ai trouvé une ligne qui commence par " une ".
J'ai trouvé une ligne qui commence par un chiffre.
J'ai trouvé une ligne qui commence par un chiffre.
J'ai trouvé une ligne qui commence par un chiffre.
Fin de la lecture du fichier !
```

Comme n'importe quel langage interprété, un script **awk** peut être exécuté directement, en le rendant exécutable et en ajoutant un *shebang* à son entête :

Fichier
```
#!/usr/bin/awk -f
BEGIN { print "Lecture d'un fichier avec awk" }
[...]
```

On peut alors l'utiliser de la manière suivante :

Terminal
```
$ ./commandes.awk exemple.txt
Lecture d'un fichier avec awk
[...]
```

2.4 Variables

Comme tout langage de programmation, **awk** permet de définir des variables afin d'y stocker des données. Cela se fait de la manière la plus simple que l'on connaisse :

Fichier
```
<nom de variable> = <valeur>
```

Par exemple :

Fichier
```
monTexte = "Salut le monde !"
```

Pour ensuite utiliser le contenu de la variable, il suffit d'en donner le nom, par exemple :

Fichier
```
print monTexte
```

Il existe également des variables spéciales, qui permettent d'accéder aux différents champs d'une ligne. Comme on l'a vu plus haut, **awk** peut facilement traiter une ligne comme un ensemble de champs, séparés par un séparateur (que l'on peut modifier avec l'option **-F**). On appelle alors simplement chaque champ par son numéro, précédé du signe **$** ; par exemple, le second champ est appelé **$2**. On a également accès à la ligne complète, par la variable **$0**.

Par exemple, pour afficher le second et le troisième champ des lignes commençant par des chiffres, lorsque le séparateur est **:**, on peut exécuter la commande suivante :

Terminal
```
$ awk -F: '/^[0-9]/ { print $2, $3 }' exemple.txt
1234 25
1.5 22
2.5 13
```

Certaines variables sont prédéfinies par le système, le tableau suivant récapitule les plus importantes d'entre elles :

Définition

Le *shebang* est une instruction particulière, à placer en première ligne d'un programme, utilisable pour tout langage interprété (donc non compilé) qui accepte le dièse (#), dont la syntaxe est la suivante :

`#!<nom de commande>`

Lorsque l'on exécute directement un tel programme, l'ordinateur reconnaît le shebang et exécute la commande indiquée, en lui donnant le fichier lui-même en argument. Pour que cela fonctionne, il ne faut pas oublier de rendre le fichier exécutable :

`chmod a+x <nom du fichier>`

À retenir

Comme on le constate dans les exemples de chaque fonction, aucune d'entre elles n'affiche quelque chose à l'écran. En effet, pour afficher quelque chose (ou plutôt pour retourner quelque chose sur la sortie standard), il faut systématiquement utiliser `print`. On retrouve le fonctionnement habituel des langages de programmation, là où `sed` affiche par défaut toute ligne qu'il a reçue en entrée, qu'elle soit traitée ou non.

Variable	Description
FS	Caractère utilisé comme séparateur de champs, en entrée (par défaut, une ou plusieurs espaces)
RS	Caractère utilisé comme séparateur de lignes, en entrée (par défaut, le retour à la ligne \n)
OFS	Caractère utilisé comme séparateur de champs, en sortie (par défaut, une espace)
ORS	Caractère utilisé comme séparateur de lignes, en sortie (par défaut, le retour à la ligne \n)
NF	Nombre de champs présents dans la ligne courante
NR	Numéro de la ligne courante

2.5 Fonctions

Le langage **awk** est composé de nombreuses fonctions, on ne pourra pas toutes les lister ici (certains livres font ça très bien), voyons-en quelques-unes...

Fonction	Description

int(x)	Retourne la partie entière du nombre x
	Exemple :
	`$ echo '2.5' | awk '{ print int($0) }'`
	`2`

gsub(r,s,t)	Remplace toute occurrence de r par s dans la chaîne t. Si on omet t, c'est toute la ligne ($0) qui est visée. La fonction retourne le nombre de fois où la substitution est effectuée.
	Exemple :
	`$ echo 'Bonjour le monde' | awk '{ gsub(/le monde/, "à toi"); print $0 }'`
	`Bonjour à toi`

sub(r,s,t)	Remplace la première occurrence de r par s dans la chaîne t. Si on omet t, c'est toute la ligne ($0) qui est visée. La fonction retourne le nombre de substitutions effectuées, donc soit 1 soit 0.

length(s)	Retourne la longueur de la chaîne s. Si on omet s, c'est toute la ligne ($0) qui est visée.
	Exemple :
	`$ echo 'Bonjour le monde' | awk '{ print "Longueur avant:", length; sub(/le monde/, "à toi"); print "Après substitution:", length }'`
	`Longueur avant: 16`
	`Après substitution: 13`

substr(s,i,n)	Retourne la sous-chaîne de la chaîne s, en commençant par le caractère i et de taille n. Si n est omis, retourne à partir de i jusqu'à la fin de la chaîne. Exemple :
	`$ echo 'Bonjour le monde' | awk '{ print substr($0, 4, 7) }'`
	`jour le`

CONCLUSION

Nous avons sommairement abordé les deux formidables outils que sont **sed** et **awk**. Bien sûr, on n'utilise généralement pas ces commandes au quotidien, mais elles sont bonnes à connaître pour les fois où on a besoin de faire des traitements automatisables sur des fichiers texte : elles permettent de gagner de précieuses minutes, parfois même de précieuses heures ! ■

AUTOMATISER DES TÂCHES RÉCURRENTES AVEC CRON

Lorsque l'on souhaite effectuer une action à heure fixe (ou à date fixe), on a deux possibilités : mettre en place une alarme sur son smartphone (ou sur son réveil) ou utiliser cron ; mais comme dit si bien l'agent Smith, « N'envoyez jamais un humain faire le travail d'un programme »...

Le système **cron** est présent dans presque tous les systèmes UNIX, il y est généralement installé par défaut ; c'est le cas sur toutes les distributions Linux. Il n'y a donc pas besoin d'installer quoi que ce soit pour utiliser **cron**, il est d'ailleurs déjà en service sur votre système.

Dans cet article, on se penchera sur le fonctionnement de **cron** sur la distribution Debian et ses dérivées (dont Ubuntu).

1. DAEMON, FICHIERS ET COMMANDE

Cron est composé de plusieurs éléments. Il y a tout d'abord le *daemon* **cron** lui-même, qui tourne en continu sur le système et qui se charge d'exécuter périodiquement les commandes qu'on lui a indiquées. On rencontre également la commande **crontab**, dont l'objectif est d'éditer les instructions données à **cron**.

Avec **crontab**, on peut modifier les instructions liées à des utilisateurs : chaque utilisateur a accès à sa propre liste d'instructions, y compris **root**. Mais il existe également des instructions globales au système, configurées dans le fichier **/etc/crontab**, ou dans les répertoires **/etc/cron.d**, **/etc/cron.daily**, **/etc/cron.weekly** et **/etc/cron.monthly**. Nous verrons comment se servir de chacune de ces méthodes.

Les instructions données à **cron** sont appelées *cronjobs*.

1.1 anacron

Ces dernières années, les distributions Linux se sont adaptées aux utilisations des postes de travail. Pour **cron**, il a fallu prendre en compte le fait que les ordinateurs peuvent être éteints la nuit : certaines tâches de « nettoyage » du système étant programmées la nuit, si l'ordinateur est arrêté toutes les nuits ces tâches ne s'exécutent jamais.

C'est pourquoi le programme **anacron** a été créé : celui-ci s'assure que les tâches quotidiennes ont été exécutées et, si l'ordinateur était éteint à l'heure concernée, la tâche s'exécute plus tard.

Sur les distributions basées sur Debian, l'utilisation d'**anacron** est très simple : lorsqu'il est installé, il gère lui-même les tâches système quotidiennes, hebdomadaires et mensuelles. Et lorsqu'il n'est pas installé, c'est **cron** qui gère ces tâches. Il n'est pas assuré que le comportement soit le même sur les autres distributions.

anacron a certains inconvénients, notamment le fait qu'il ne sait que gérer les périodicités quotidienne, hebdomadaire et mensuelle... Mais il répond correctement au besoin de base que l'on a évoqué : exécuter les tâches nocturnes lorsque l'on allume l'ordinateur le matin.

2. SYNTAXE D'UN CRONJOB

Un cronjob est simplement une ligne dans un fichier de configuration. Cette ligne est composée de 6 champs, séparés par des espaces. Dans l'ordre, ces champs sont :

- la minute à laquelle exécuter la tâche (entre **0** et **59**) ;
- l'heure à laquelle exécuter la tâche (entre **0** et **23**) ;
- le jour du mois auquel exécuter la tâche (entre **1** et **31**) ;
- le mois auquel exécuter la tâche (entre **1** et **12**) ;
- le jour de la semaine auquel exécuter la tâche (entre **0** et **7**, ces deux valeurs représentant le dimanche) ;
- la commande à exécuter (ce champ peut contenir des espaces).

À savoir

Le nom du service cron provient simplement du dieu grec du temps, Chronos. Il n'y a parfois pas à chercher bien loin.

Son histoire, par contre, est plus compliquée. Une première version de cron, assez limitée, est apparue à la fin des années 70 dans UNIX V7 ; celle-ci était particulièrement gourmande en ressources et ne pouvait être utilisée que par le super-utilisateur. C'est avec UNIX System V, au début des années 80, qu'est apparue une implémentation capable de supporter des systèmes multi-utilisateurs sans surcharge. L'implémentation de cron que l'on utilise généralement de nos jours est apparue à la fin des années 80 et a évolué au fur et à mesure...

Pour les cinq premiers champs, on peut remplacer la valeur numérique par un astérisque * pour indiquer que la tâche est à exécuter à chaque fois (par exemple, un * dans le troisième champ indique que la tâche doit s'exécuter tous les jours).

Par exemple, exécuter la commande **sync** tous les mardis à 14h23 (tous les jours du mois, tous les mois de l'année) se traduira par ;

Fichier
```
23  14  *  *  2  sync
```

2.1 Valeurs multiples

On peut programmer des cronjobs de manière un peu plus évoluée, avec une syntaxe particulière dans les cinq premiers champs :

- On peut définir plusieurs valeurs séparées par des virgules : **10,22,45** dans le premier champ résultera en l'exécution de la commande à la 10e, 22e et 45e minute ;
- On peut également définir des intervalles, en séparant le début et la fin par un tiret : **9-18** dans le second champ résultera en l'exécution de la commande entre 9h et 18h ;
- Enfin, on peut définir des récurrences (« tous les X temps »), avec la syntaxe ***/X** : ***/3** dans le troisième champ résultera en l'exécution de la commande tous les 3 jours.

Notons que ces syntaxes peuvent être combinées.

Exemple : exécuter la commande **date >> /tmp/la_date** toutes les 5 minutes, de 20h à 4h55 du matin :

Fichier
```
*/5  20-23,0-4  *  *  *  date >> /tmp/la_date
```

2.2 Syntaxe pour les cronjobs système

Lorsque l'on met en place un cronjob système dans le fichier **/etc/crontab** ou dans un fichier du répertoire **/etc/cron.d**, le système ne sait pas quel utilisateur doit être utilisé pour lancer la commande : il faut le lui indiquer dans un champ supplémentaire, placé en sixième position ; la commande correspond alors au septième champ de la ligne.

3. CRÉER OU MODIFIER DES CRONJOBS

Il existe trois méthodes pour créer des tâches **cron**...

3.1 Éditer les tâches d'utilisateur

La façon la plus connue d'éditer des tâches est la commande **crontab**. Celle-ci donne accès à la liste des tâches créées par l'utilisateur qui l'exécute.

À savoir

Si vous souhaitez plus d'informations sur cron, n'hésitez pas à consulter les pages de manuel : man cron pour en savoir plus sur le daemon, man crontab pour en savoir plus sur la commande crontab elle-même et man 5 crontab pour en savoir plus sur la syntaxe.

Pour éditer les tâches **cron** d'un utilisateur, la commande à utiliser est :

Terminal
```
crontab -e
```

Cette commande exécutera l'éditeur de texte par défaut, celui-ci éditant un fichier temporaire qui sera placé dans le répertoire système idoine lors de la fermeture de l'éditeur.

C'est dans ce fichier temporaire que l'on place les lignes évoquées ci-dessus. Lors de la fermeture de l'éditeur, **crontab** vérifie la validité syntaxique de ces lignes.

On peut vouloir choisir l'éditeur à utiliser, par exemple **emacs** ; dans ce cas, la commande devient :

Terminal
```
EDITOR=emacs crontab -e
```

Pour visualiser les tâches **cron** d'un utilisateur, la commande est :

Terminal
```
crontab -l
```

3.2 Éditer les tâches système

Les tâches système, quant à elles, peuvent être placées dans deux endroits différents :

- ⇨ soit dans le fichier **/etc/crontab** ;
- ⇨ soit dans un fichier (dont le choix du nom est libre) à placer dans le répertoire **/etc/cron.d**.

Dans les deux cas, il faut préciser le nom de l'utilisateur dans le 6e champ, comme indiqué plus haut.

Exemple : exécuter **wget http://www.example.com/exemple.html -O /srv/www/remote/exemple.html** en tant que **www-data**, tous les deux jours à minuit pile :

Fichier
```
0  0  */2  *  *  www-data  wget http://www.example.com/exemple.html -O
/srv/www/remote/exemple.html
```

Attention : avec ces fichiers, la commande **crontab** n'est pas là pour vérifier la validité syntaxique ; dans le cas d'une erreur, on ne le verrait qu'un peu plus tard, dans les journaux du système...

3.3 Tâches système à récurrence simple

Dans le cadre des tâches système, on a aussi accès à quatre répertoires particuliers. Dans ceux-ci, il suffit de placer un exécutable (script ou autre), ou un lien vers un exécutable, pour qu'il soit automatiquement lancé avec une périodicité donnée – on a moins de maîtrise sur le moment précis de l'exécution et on ne peut pas choisir l'utilisateur (c'est systématiquement **root**), mais on n'a pas à gérer une ligne particulière, dans laquelle on pourrait faire des erreurs de syntaxe. Il s'agit des répertoires :

- ⇨ **/etc/cron.hourly** : commandes à exécuter toutes les heures ;
- ⇨ **/etc/cron.daily** : commandes à exécuter tous les jours ;

- **/etc/cron.weekly** : commandes à exécuter toutes les semaines ;
- **/etc/cron.monthly** : commandes à exécuter tous les mois.

Le lancement de ces exécutables est géré par **anacron** ou par **cron**, grâce aux lignes suivantes du fichier **/etc/crontab** :

Fichier
```
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-
parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-
parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-
parts --report /etc/cron.monthly )
```

À savoir

Les cronjobs utilisateurs permettent à chacun d'éditer ses propres tâches ; ils sont parfaitement adaptés aux besoins des utilisateurs (par exemple s'envoyer un e-mail tous les jours, contenant des données particulières), qui peuvent les gérer de manière indépendante, sans risquer de causer une défaillance de cron (la syntaxe étant vérifiée par crontab).

Les tâches système sont réunies dans un endroit cohérent, où on peut répartir les tâches dans plusieurs fichiers de configuration et centraliser les tâches de différents utilisateurs. C'est notamment à utiliser dans le cadre d'une administration de serveur, où le concept d'utilisateur est réduit et où il est plus simple de centraliser la configuration dans /etc, plutôt que d'utiliser la commande crontab en tant que plusieurs utilisateurs différents.

4. ENVOI DE MESSAGES

Par défaut, **cron** envoie un e-mail à la personne concernée avec le retour de la commande exécutée ; cela permet d'avoir confirmation de son bon fonctionnement. Pour qu'une commande soit exécutée sans envoyer d'e-mail, il suffit de faire en sorte qu'elle ne retourne rien, par exemple en ajoutant une redirection vers **/dev/null** à la fin de la commande, comme ceci :

Fichier
```
0 0 * * * date >/dev/null
```

(Cet exemple n'a bien évidemment aucune utilité).

5. TÂCHES PONCTUELLES AVEC AT

Le système **cron** permet de gérer des tâches récurrentes : il n'est pas du tout adapté à l'exécution ponctuelle d'une commande à une date/heure donnée ; pour une exécution ponctuelle, on utilise la commande **at**.

Cette commande utilise une syntaxe très simple : elle prend la commande à exécuter sur son entrée standard et une définition de temps en argument.

La définition de temps peut être :

- une heure précise, sous la forme **HH:MM**, par exemple **16:23** ;
- un délai à partir de « maintenant » (*now*), avec la syntaxe **now + <durée>**, par exemple **now + 1 hours** ; la durée est composée de minutes (**minutes**), d'heures (**hours**), de jours (**days**) ou de semaines (**weeks**).

Par exemple, pour lancer le téléchargement d'un fichier dans la nuit, on peut exécuter :

Terminal
```
$ echo "wget http://www.example.com/gros_fichier.tar.
gz" | at 01:30
warning: commands will be executed using /bin/sh
job 1 at Fri Mar 21 01:30:00 2014
```

at nous informe que les commandes seront exécutées avec le shell **/bin/sh** et rappelle la date précise à laquelle la commande sera exécutée.

Enfin, on peut consulter les tâches en attente avec la commande **atq** et en supprimer avec la commande **atrm**.

Notons qu'**at** n'est pas nécessairement installé par défaut : il faut alors installer le paquet **at**.

6. AUTOMATISER LES MISES À JOUR

Commençons cette section par un avertissement : l'automatisation des mises à jour n'est pas nécessairement une bonne idée. Cependant, on peut au moins automatiser leur téléchargement !

Pour automatiser ces aspects, nul besoin de réfléchir à la commande précise que l'on doit utiliser : Debian et Ubuntu proposent le paquet **cron-apt**, qui sert précisément à cela.

Lors de l'installation de ce paquet, le fichier **/etc/cron.d/cron-apt** se met en place ; celui-ci définit la fréquence à laquelle effectuer les opérations sur les paquets. La configuration par défaut lance ce logiciel tous les jours à 4h du matin :

Fichier

```
0 4    * * *   root    test -x /usr/sbin/cron-apt && /usr/sbin/cron-apt
```

Le paramétrage de **cron-apt** se trouve dans le répertoire **/etc/cron-apt**. Le fonctionnement de ce logiciel est très simple : il prend le contenu des fichiers présents dans **/etc/cron-apt/action.d** et utilise chaque ligne comme argument à la commande **apt-get**. Par défaut, on retrouve deux fichiers dans ce répertoire :

⇨ **/etc/cron-apt/action.d/0-update** :

Fichier

```
update -o quiet=2
```

⇨ **/etc/cron-apt/action.d/3-download** :

Fichier

```
autoclean -y
dist-upgrade -d -y -o APT::Get::Show-Upgraded=true
```

Par conséquent, avec sa configuration par défaut, **cron-apt** exécute les commandes suivantes tous les jours à 4h du matin :

Terminal

```
apt-get update -o quiet=2
apt-get autoclean -y
apt-get dist-upgrade -d -y -o APT::Get::Show-Upgraded=true
```

De cette manière, les mises à jour sont quotidiennement téléchargées, mais non appliquées.

Chacun sera libre de modifier ces fichiers ou d'en ajouter un dans **/etc/cron-apt/action.d**, par exemple pour appliquer automatiquement les mises à jour (avec les risques que cette opération comporte)...

Notons enfin qu'il peut être utile de lire le contenu du fichier **/usr/share/doc/cron-apt/README.gz** pour en savoir plus sur ce logiciel. ■

1 LES CLASSIQUES

SYNCHRONISER SES DONNÉES AVEC RSYNC

Pour différentes raisons, on souhaite synchroniser des données d'un espace à un autre, souvent sur deux ordinateurs différents. Pour cela, on peut copier manuellement des fichiers, mais c'est très contraignant. Il existe en réalité différents logiciels capables de synchroniser des données, rsync étant le plus courant de ceux-là.

La synchronisation de données permet beaucoup de choses : travailler sur deux ordinateurs différents, maintenir des données identiques d'un client vers un serveur, voire entre deux serveurs, mais également, dans une moindre mesure, sauvegarder des données. Pour toutes ces actions, on imagine généralement des manipulations utilisant des outils simples (comme `cp`), mais très contraignantes (toute vérification doit être faite manuellement, l'intégralité des données est à transmettre à chaque fois, etc.), alors que des outils dédiés existent et sont simples à utiliser.

Comme son nom l'indique, `rsync` est un logiciel de **synchronisation à distance** ; il fonctionne de manière **unidirectionnelle** : les fichiers vont d'un ordinateur A à un ordinateur B. Si on veut transférer dans l'autre sens, il faut exécuter `rsync` une seconde fois (et veiller à ce qu'il n'y ait pas de conflit, par exemple des fichiers qui ont été modifiés sur les deux ordinateurs).

Il existe également `unison`, un outil de synchronisation **bidirectionnelle**, qui gère les modifications des deux côtés et propose de manière interactive des solutions dans le cadre de la résolution de conflits de modifications. `rsync` est plus adapté aux synchronisations de fichiers dans une seule direction, dans le cadre d'une mise en production, d'une récupération de données distantes ou de sauvegardes par exemple, surtout lorsque ces manipulations sont à automatiser.

Pour les transferts par le réseau, `rsync` s'appuie par défaut sur la commande `rsh`, proposée notamment par OpenSSH, c'est donc le protocole SSH qui est utilisé ; il faut donc qu'un serveur SSH soit installé sur la machine distante. Dans le cadre de cet article, nous estimerons que ce serveur SSH est déjà installé et qu'une authentification par clé SSH sans phrase de passe est en place, permettant d'initier ces transferts sans taper de mot de passe, les rendant ainsi aisément automatisables.

Note

`rsync` (pour *remote synchronization*) existe depuis 1996. Il a été développé dans le cadre de travaux de recherche sur un protocole de compression « longue distance », capable de détecter des redondances (et donc, d'améliorer la compression de certaines données), par Andrew Tridgell, développeur australien qui fut notamment le premier développeur de Samba.

1. INSTALLATION

`rsync` étant un logiciel très courant, il est disponible dans les sources de quasiment toutes les distributions Linux, ainsi que dans celles d'autres systèmes d'exploitation de type UNIX comme les BSD. Sous Windows, il existe cwRsync (*Cygwin Rsync*).

Dans notre cas, nous utiliserons `rsync` sur un poste de travail Ubuntu et un serveur Debian. Il est installé (sur les deux machines) avec cette simple commande, précédée de `sudo` si nécessaire :

Terminal

```
apt-get install rsync
```

Bien sûr, les manipulations avec `rsync` seront les mêmes quelle que soit la distribution Linux utilisée, et que les transferts se fassent entre un client et un serveur ou entre deux serveurs.

2. PRINCIPE DE FONCTIONNEMENT

Le travail de **rsync** est d'observer le contenu du répertoire à transférer et de sa cible, puis d'analyser les fichiers qui s'y trouvent. S'il détecte une différence quelconque entre les fichiers de la source et ceux de la destination, il remplace les seconds par les premiers.

Ceci ne concerne pas que l'existence ou non de fichiers, mais également leur contenu. Pour analyser et comparer les contenus de fichiers distants, **rsync** utilise un algorithme de recherche par somme de contrôle (*checksum*) : ce n'est pas le contenu des fichiers qui est comparé, mais leur empreinte, bien plus légère. On retrouve en réalité ce type de contrôle partout. Par exemple, le numéro à 16 chiffres d'une carte bancaire comporte deux systèmes de vérification de ce type, qui permettent d'assurer la validité de la carte. **rsync** utilise le même principe : il calcule pour chaque fichier une somme de contrôle sur la source et la destination, puis il transfère le fichier uniquement si celui-ci a changé. De plus, il sait travailler sur des portions de fichier : c'est ce qui permet de maintenir synchronisés deux fichiers de grande taille sans avoir à transférer tout le fichier à chaque modification.

La syntaxe d'utilisation de **rsync** est la suivante :

Fichier

```
rsync <options> <source> <destination>
```

La commande **rsync** accepte différents arguments et options, nous utiliserons les suivants :

- **-a** (*archive*) : cette option permet de demander à **rsync** de conserver toutes les informations des fichiers : propriétaire, date de modification, etc. ;
- **-v** (*verbose*) : avec cette option, on demande à **rsync** de donner des indications sur les opérations qu'il effectue ;
- **--stats** : lorsque l'on donne cet argument à **rsync**, il retourne en fin d'exécution des statistiques sur les transferts effectués ;
- **--delete** : cette option force **rsync** à supprimer, sur la destination, les fichiers qui ont été supprimés sur la source – en son absence, tous les fichiers sont conservés ;
- **--backup** : cette option demande à **rsync** de conserver, sur le serveur distant, des sauvegardes des anciennes versions des fichiers sauvegardés – elle est associée à l'option **--backupdir** pour indiquer où conserver ces anciennes versions.

3. UTILISATION DE RSYNC

On étudiera ici trois façons d'utiliser **rsync** :

- Synchroniser des données entre deux répertoires d'une même machine ;
- Synchroniser des données d'un répertoire local vers un serveur distant ;
- Synchroniser des données d'un serveur distant vers un répertoire local.

Nous verrons également comment maintenir des sauvegardes des anciennes données, afin de pouvoir retourner en arrière si nécessaire.

3.1 Synchronisation locale

Dans cet exemple, nous allons synchroniser à partir du répertoire **/tmp/ma_source** vers le répertoire **/tmp/ma_destination**, sur une même machine. L'utilisation de **rsync** (avec les options listées au point 2) est alors très simple :

Terminal
```
rsync -av --stats --delete /tmp/ma_source /tmp/ma_destination
```

Cette commande peut être effectuée autant de fois que l'on veut : seules les données non synchronisées seront copiées lors de chaque exécution successive. Notons que les fichiers supprimés sur la source seront aussi supprimés sur la destination : assurez-vous que c'est bien le comportement que vous souhaitez !

3.2 Synchronisation vers un serveur distant

En termes de commande à exécuter, une synchronisation vers un serveur distant n'est pas très différente d'une synchronisation locale : il ne faut en réalité que renseigner le nom du serveur en préfixe de la destination.

Dans cet exemple, la destination est le répertoire **/tmp/ma_destination** sur le serveur **machine_distante** :

Terminal
```
rsync -av --stats --delete /tmp/ma_source machine_distante:/tmp/ma_
destination
```

3.3 Synchronisation à partir d'un serveur distant

Cette fois-ci, c'est la source qui est sur le serveur **machine_distante**, les noms des répertoires restant les mêmes :

Terminal
```
rsync -av --stats --delete machine_distante:/tmp/ma_source /tmp/ma_
destination
```

Comme on le constate, la syntaxe reste particulièrement simple et il n'y a aucune spécificité liée à l'emplacement de la source ou de la destination.

Notons toutefois qu'on ne peut pas transférer d'une machine distante vers une autre machine distante : **rsync** n'acceptera pas de nom de machine en préfixe des deux arguments.

3.4 Sauvegardes

Lors d'un transfert de fichiers, la destination est modifiée conformément à ce qui est trouvé dans la source. Afin de garder trace des anciennes versions des fichiers, on peut demander à **rsync** d'en faire des sauvegardes avant modification ou effacement. Cela se fait systématiquement du côté de la destination. Cela veut dire que dans le cas d'une

synchronisation d'un client vers un serveur, c'est sur le serveur que les sauvegardes sont effectuées – et dans le cas d'une synchronisation du serveur vers le client, c'est sur le client que les sauvegardes sont effectuées.

Pour mettre en place de telles sauvegardes, il faut ajouter les arguments suivants à la commande :

Terminal
```
--backup --backup-dir=/répertoire/des/sauvegardes
```

Les fichiers modifiés sont alors sauvegardés dans le répertoire concerné.

Pour conserver un historique jour après jour, on peut demander d'effectuer des sauvegardes dans un répertoire nommé d'après la date du jour, en utilisant la commande **date**. Une utilisation simple de cette commande pour obtenir la date du jour peut être la suivante (ici, on obtient le 25 mars 2014) :

Terminal
```
$ date +'%Y-%m-%d'
2014-03-25
```

Les arguments pour **rsync** deviennent alors par exemple :

Terminal
```
--backup --backup-dir=/répertoire/des/sauvegardes/$(date +'%Y-%m-%d')
```

3.5 Exemple concret

Imaginons que l'on souhaite sauvegarder quotidiennement le répertoire **/srv/www** d'un serveur appelé *prod*, en plaçant ces sauvegardes dans le répertoire **/data/sauvegarde/prod-www/actuel** d'un serveur appelé *sauvegarde* ; on veut conserver les archives des données modifiées dans un répertoire **/data/sauvegarde/prod-www/<date du jour>**. Et on veut effectuer cela en exécutant la commande **rsync** sur le serveur *sauvegarde*.

On peut alors créer le script suivant :

Fichier
```
#!/bin/sh
SOURCE="prod:/srv/www"
ACTUEL="/data/sauvegarde/prod-www/actuel"
BACKUP="/data/sauvegarde/prod-www/$(date +'%Y-%m-%d')"
mkdir -p "$BACKUP"
/usr/bin/rsync -av --stats --delete --backup --backup-dir="$BACKUP"
"$SOURCE" "$ACTUEL"
rmdir --ignore-fail-on-non-empty $BACKUP
```

Ce script peut alors être exécuté automatiquement toutes les nuits grâce à **cron**.
Son fonctionnement est le suivant :

- On place dans les variables **SOURCE**, **ACTUEL** et **BACKUP** les chemins des différents éléments traités ;
- On crée le répertoire qui accueillera les sauvegardes ;
- On effectue la synchronisation ;
- On efface le répertoire des sauvegardes s'il est vide.

4. ALLER PLUS LOIN

Le commandes présentées ici peuvent toutes être automatisées par le biais d'un script similaire à l'exemple étudié à l'instant, exécuté par `cron`. En termes d'extension possible, il est relativement difficile de pousser plus loin, étant donné que `rsync` fait simplement le travail qu'on lui demande, sans fioritures. On pourrait cependant travailler autour de l'option `-e`, qui permet de fournir une commande spéciale pour se connecter à distance (par exemple se connecter en SSH sur un port spécifique, ou avec un nom d'utilisateur particulier), mais il n'y a pas grand-chose de plus à voir.

Si on a des besoins plus poussés en synchronisation ou en sauvegarde, on peut alors se tourner vers d'autres logiciels...

On a déjà abordé la question de la synchronisation bidirectionnelle avec **Unison** : ce logiciel est également le fruit d'un travail de recherche. Il peut également utiliser une connexion SSH pour synchroniser des espaces de stockage distants. Sur un sujet assez similaire, de nos jours la mode est au « cloud » : de nombreuses sociétés offrent des espaces gratuits, que l'on peut étendre moyennant le paiement d'un abonnement. Il existe bien sûr des logiciels libres permettant de reproduire ces comportements. L'un des plus connus est **ownCloud**, il est malheureusement très peu orienté « ligne de commandes » et la synchronisation de fichiers y est une fonctionnalité parmi de nombreuses autres. On peut alors compter sur plusieurs « outsiders » utilisables totalement en ligne de commandes :

- **Syncany** : encore en développement, ce logiciel peut utiliser n'importe quel protocole de partage de fichiers pour effectuer ses synchronisations, par ailleurs tous les fichiers sont chiffrés avant envoi ;
- **Seafile** se base sur un protocole spécifique, inspiré de Git, pour gérer les synchronisations multidirectionnelles – il nécessite l'installation d'un serveur qui lui est propre afin de fonctionner, mais selon votre besoin, le jeu peut en valoir la chandelle ;
- **SparkleShare** utilise un dépôt Git comme point central : il peut alors bénéficier des avantages de Git, il peut être hébergé sur de nombreux serveurs (par exemple sur GitHub), par contre il souffre des limites de Git : il n'est pas adapté pour la gestion de gros volumes (musique, vidéo), ni de gros fichiers changeants : il est extrêmement gourmand en espace disque (compter au moins le double de l'espace utilisé par les fichiers).

Sur le sujet de la sauvegarde, il existe de nombreux logiciels qui répondent à ce besoin. Rien qu'en restant autour de `rsync`, plusieurs solutions se présentent, notamment `rdiff-backup` et `rsnapshot` : ces deux logiciels sont assez similaires sur le principe, ils utilisent `rsync` pour le transfert de fichiers entre l'espace de travail et l'espace de sauvegarde. `rdiff-backup` donne accès à la dernière version de la sauvegarde et permet de restaurer les précédentes grâce à des fichiers contenant les différences entre la dernière version et la précédente, il est alors très économe en espace disque ; `rsnapshot`, de son côté, permet d'accéder directement aux différentes versions des sauvegardes, il utilise ses liens « hard » afin de limiter la place utilisée en faisant pointer plusieurs sauvegardes vers un même fichier si celui-ci n'a pas changé – par contre, au moindre changement dans un fichier, `rsnapshot` stocke celui-ci en deux exemplaires : il est alors plus gourmand en espace disque.

Enfin, on ne peut pas parler de sauvegarde sans évoquer **Bacula**, un outil complet de sauvegarde en réseau : tout à fait adapté lorsque l'on a un parc complet à sauvegarder, il est toutefois excessif s'il s'agit d'effectuer les sauvegardes d'une seule machine. ■

SURVEILLER LES MODIFICATIONS AU SEIN DES FICHIERS

De nombreux fichiers d'un système d'exploitation changent régulièrement. D'autres fichiers ont un rythme de changement bien plus réduit, généralement uniquement lors de mises à jour. Si de tels fichiers sont modifiés en dehors d'une mise à jour, cela peut devenir inquiétant...

Tous les fichiers d'un système n'ont pas la même importance. Nous pouvons distinguer plusieurs cas :

- Les fichiers dont les changements sont courants et normaux, qu'il n'y a pas lieu de surveiller : les journaux système dans `/var/log`, les fichiers nécessaires au fonctionnement d'applications dans `/var/run`, et de manière générale tout fichier dans `/var` – on peut ajouter également le répertoire `/tmp` et son contenu ;
- Les fichiers appartenant aux utilisateurs, placés dans leurs répertoires personnels, qu'un administrateur ne maîtrise pas ;
- Les fichiers de configuration, généralement placés dans `/etc`, qui ne sont modifiés que par l'administrateur ;
- Les fichiers système, dans `/bin` ou `/usr/bin` par exemple, qui ne devraient changer que lors d'une mise à jour.

Les deux dernières catégories regroupent des fichiers qui peuvent être modifiés lorsqu'un système est compromis : cela permet à un attaquant de « rester » dans un système qu'il a compromis, en plaçant judicieusement des bouts de code dans certains fichiers, lui permettant de rester indétectable (ses modifications ayant notamment pour but de cacher sa présence à l'administrateur du système). On appelle cela un *rootkit*.

La méthode que l'on abordera ici permet de surveiller le système pour détecter de telles modifications, nous permettant ainsi de mieux maîtriser des outils système de base. Notons que nous exécutons les commandes directement avec l'utilisateur *root*, certains fichiers système étant inaccessibles aux utilisateurs classiques ; pour faire des tests dans un terminal sous Ubuntu par exemple, il faut ajouter `sudo` devant les commandes.

1. MÉTHODE DE VÉRIFICATION

Afin de vérifier des changements dans un fichier, on utilisera la commande `md5sum` : son objectif est de générer une empreinte MD5 (*hash*) d'un fichier : c'est une chaîne de caractères courte générée à partir d'un flux de données (contenu d'un fichier ou autre), qui permet de l'identifier de manière unique.

Cette commande s'utilise de la manière suivante :

Terminal
```
md5sum <nom(s) de fichier(s)>
```

Elle retourne une ligne par fichier, contenant le hash et le nom du fichier.

Note

Cet article permet de découvrir et d'utiliser plusieurs commandes (`find`, `sed`, `md5sum`...) et astuces des *shells* (changement de couleur d'affichage, par exemple).

À savoir

Bien sûr, en générant une chaîne courte à partir d'un gros fichier, on n'est pas à l'abri de collisions : deux fichiers différents peuvent générer le même *hash*. Mais il faut que ces deux fichiers soient très différents, les modifications qu'un pirate mettrait en place ne seraient alors pas vraiment discrètes (le comportement du logiciel ne serait plus du tout le même). Cela étant dit, rien n'est infaillible en informatique et un pirate très expérimenté pourrait être capable de générer une situation qui lui serait favorable ; mais un tel déploiement d'ingéniosité est-il proportionnel à la sensibilité de vos systèmes ?

Par exemple :

Terminal
```
# md5sum /etc/passwd /etc/group
41794663561df2b7eeae267948a280e3  /etc/passwd
0b9a75cb1d1d0113561d3cb5139c6f7c  /etc/group
```

1.1 Liste des fichiers

Comme on le voit, on donne les noms des fichiers directement comme arguments à la commande **md5sum**. Mais on ne va pas entrer à la main la liste des fichiers à analyser ! On utilise alors la commande **find** afin de lister ces fichiers.

La commande **find** s'utilise de la manière suivante :

Terminal
```
find <chemin> <critères>
```

Par exemple :

Terminal
```
# find /etc -type f
[...]
/etc/group
[...]
/etc/passwd
[...]
```

On liste ici tous les fichiers simples (type **f**) dans le répertoire **/etc** et ses descendants.

On peut alors combiner ces deux commandes de la manière suivante :

Terminal
```
# find /etc -type f -exec md5sum {} \+
[...]
0b9a75cb1d1d0113561d3cb5139c6f7c  /etc/group
[...]
41794663561df2b7eeae267948a280e3  /etc/passwd
[...]
```

À retenir

En général, on termine l'argument `-exec` de `find` par les caractères `\;`. Cela permet d'exécuter la commande indiquée une fois pour chaque fichier trouvé. Avec `\+`, on dit à `find` de concaténer plusieurs fichiers en argument à la commande donnée, cela permet d'exécuter beaucoup moins d'instances de `md5sum`.

1.2 Stockage de la liste des fichiers

Afin de comparer une empreinte avec la précédente, il est nécessaire de stocker la précédente. On fait cela par une redirection dans un fichier, avec le caractère **>**.

Profitons-en pour lister l'ensemble des fichiers que l'on veut surveiller, c'est-à-dire les fichiers contenus dans **/etc** , **/usr/bin**, **/usr/sbin**, **/usr/lib**, **/bin**, **/sbin** et **/lib**. On stocke ici cette liste dans le fichier **/tmp/md5**.

Terminal
```
# find /etc /usr -type f -exec md5sum {} \+ > /tmp/md5
```

Cette commande peut mettre beaucoup de temps à se terminer (parfois plus d'une minute), ne vous inquiétez pas...

2. COMPARAISON

Une fois la liste des empreintes stockée une première fois, on peut l'utiliser pour vérifier si des fichiers ont été changés. Cela se fait avec l'argument **-c** (pour *check*) de **md5sum** :

Terminal
```
md5sum -c <fichier des empreintes>
```

Cette commande retourne une ligne pour chaque fichier vérifié. Si l'on ne veut afficher que les fichiers en erreur, on peut ajouter l'argument **--quiet**.

On obtiendra alors par exemple :

Terminal
```
# md5sum -c /tmp/md5 --quiet
/etc/passwd- : ÉCHEC d'ouverture ou de lecture.
/etc/passwd: ÉCHEC
md5sum: AVERTISSEMENT : les sommes de contrôle 2 ne concorde pas
```

Notez que la dernière ligne est une erreur de la part des traducteurs, elle devrait dire « 2 sommes de contrôle ne concordent pas » au lieu de « les sommes de contrôle 2 ne concordent pas ».

Cet avertissement est en réalité retourné sur la sortie d'erreur de la commande (*stderr*) et non sur sa sortie standard. On peut « cacher » la sortie d'erreur si on le souhaite, avec l'opérateur de redirection **2>**. Redirigeons la sortie d'erreur vers **/dev/null** :

Terminal
```
# md5sum -c /tmp/md5 --quiet 2>/dev/null
/etc/passwd- : ÉCHEC d'ouverture ou de lecture.
/etc/passwd: ÉCHEC
```

Il est important de distinguer la sortie d'erreur de la sortie standard. En effet, quand on redirige la sortie vers un autre programme avec l'opérateur **|**, c'est uniquement la sortie standard qui est redirigée : la sortie d'erreur est perdue. On peut toutefois rediriger la sortie d'erreur vers la sortie standard avec l'opérateur **2>&1**.

2.1 Des messages plus parlants

D'accord, le message **ÉCHEC** est assez clair, il y a quelque chose qui ne va pas. Mais un logiciel facile à comprendre est un logiciel convivial, on peut alors utiliser **sed** pour changer ces messages, avec la commande suivante :

Terminal
```
sed "s/ÉCHEC d'ouverture ou de lecture./Le fichier n'existe plus/;s/:
ÉCHEC/ : Le fichier a changé/"
```

On l'intègre alors à notre recette de la manière suivante :

Terminal

```
# md5sum -c /tmp/md5 --quiet 2>/dev/null | sed "s/ÉCHEC
d'ouverture ou de lecture./Le fichier n'existe plus/;s/:
ÉCHEC/ : Le fichier a changé/"
/etc/passwd- : Le fichier n'existe plus
/etc/passwd : Le fichier a changé
```

Attention : une telle transformation limite l'utilisation de cette commande à des systèmes en français et dont les messages n'ont pas été traduits différemment.

3. COLORISATION

Et si nous mettions un peu de couleur dans les retours de ces commandes ? En effet, la sortie de la commande **md5sum** est assez basique et n'est que du texte pur. Les messages seraient visuellement plus compréhensibles s'ils étaient en couleurs.

Pour coloriser un texte dans un terminal, il faut utiliser des chaînes spéciales, qui prennent la forme suivante : **^[[<numéro>m**.

Dans cette chaîne, le texte **<numéro>** est remplacé par un numéro de style de texte :

- ⇨ **0** pour la couleur par défaut ;
- ⇨ **31** pour le rouge ;
- ⇨ **33** pour le jaune...

Par ailleurs, les caractères **^[** représentent la valeur ASCII de la touche [Échap] et non la suite de caractères « **^** » puis « **[** ». Pour obtenir une valeur ASCII d'une touche, il faut la précéder de la combinaison [Ctrl]+[V]. Il faut donc taper [Ctrl]+[V] puis [Échap]. Le second **[**, quant à lui, est bien un crochet ouvrant.

Récapitulons : si on veut changer la couleur d'un texte dans un terminal, il faut taper [Ctrl]+[V] puis [Échap], puis **[** suivi du numéro d'une couleur, puis **m**.

Notons également qu'à la fin d'un texte en couleurs, il faut rétablir le style d'origine avec **^[[0m**.

Nous allons donc changer en jaune la couleur des lignes contenant « Le fichier n'existe plus » et en rouge la couleur des lignes contenant « Le fichier a changé » et préciser qu'à la fin de chaque ligne on veut retourner sur le style par défaut, avec les instructions **sed** suivantes :

Fichier

```
s/\(.*\): ÉCHEC d'ouverture ou de lecture./^[[33m\1 : Le fichier
n'existe plus/
s/\(.*\): ÉCHEC/^[[31m\1 : Le fichier a changé/
s/$/^[[0m/
```

Il ne reste qu'à intégrer ces trois instructions **sed** en remplacement de la commande qui existe déjà :

À savoir

Chaque émulateur de terminal peut interpréter les codes couleurs de la manière qu'il le souhaite. Par exemple, Konsole affiche par défaut du texte orange au lieu du jaune. Ce comportement est généralement configurable au niveau de l'émulateur lui-même.

Terminal

```
md5sum -c /tmp/md5 --quiet 2>/dev/null | sed "s/\(.*\): ÉCHEC d'ouverture
ou de lecture./^[[33m\1 : Le fichier n'existe plus/;s/\(.*\):
ÉCHEC/^[[31m\1 : Le fichier a changé/;s/$/^[[0m/"
```

Cela donne le résultat suivant :

```
root@amaretto:~# md5sum -c /tmp/md5 --quiet 2>/dev/null | sed "s/\(.
*\): ÉCHEC d'ouverture ou de lecture./^[[33m\1 : Le fichier n'existe
 plus/;s/\(.*\): ÉCHEC/^[[31m\1 : Le fichier a changé/;s/$/^[[0m/"
/etc/passwd-  : Le fichier n'existe plus
/etc/passwd : Le fichier a changé
```

Cet usage particulier de [Ctrl]+[V] puis [Échap] peut également être remplacé par la chaîne **\033** : ce nombre est le code ASCII de la touche [Échap], en octal. Mais attention, dans ce cas il faut demander à la commande qui l'interprète (en général **echo**) d'interpréter les anti-slashes ! Les deux commandes sont alors équivalentes :

Terminal

```
echo "^[[31mGoodbye World^[[0m"
echo -e "\033[31mGoodbye World\033[0m"
```

Enfin, notons qu'il existe de nombreux codes de ce type, permettant différents rendus...
Voici une commande que vous pourrez tenter de taper dans votre terminal :

Terminal

```
echo -e "\033[31mNormal\033[0m, \033[1;31mGras\033[0m, \033[3;31mItalique\033[0m,
\033[4;31mSouligné\033[0m, \033[7;31mInversé\033[0m, \033[9;31mBarré\033[0m,
\033[1;3;31mGras italique\033[0m, \033[1;3;4;7;9;31mGras italique souligné
inversé barré (si si)...\033[0m"
```

Le résultat dépendra bien sûr de l'interprétation qui est faite de ces commandes par le terminal...

4. ALLER PLUS LOIN

On n'a abordé qu'une manière simple de traiter le problème évoqué, mais on pourrait aller beaucoup plus loin dans la mise en forme et le traitement de ces informations. On pourrait imaginer les compléments suivants :

- Refaire une base **md5sum** régulièrement (car en l'état actuel, si un fichier est ajouté, il ne sera pas surveillé) ;
- Stocker le fichier **/tmp/md5** ailleurs (car si un éventuel pirate trouve ce fichier, il aura tôt fait de remplacer les lignes qui le dérangent par l'empreinte MD5 des fichiers vérolés qu'il met en place) ;
- Utiliser un script **awk** pour compter le nombre de fichiers modifiés et le nombre de fichiers disparus ;
- Utiliser **cron** pour automatiser ces vérifications ;
- Utiliser la commande **mail** pour envoyer un rapport par e-mail...

Libre à vous d'approfondir dans ces directions, afin d'apprendre à utiliser ces commandes.

Et si vous voulez mettre une vérification de ce type sur un serveur de production, n'hésitez pas à vous tourner vers des utilitaires dédiés, comme **rkhunter** et **chkrootkit**. Ceux-ci sont un peu plus compliqués à aborder et à bien comprendre, mais dans un cadre professionnel ce choix se justifiera facilement ! ■

2

2 MATÉRIEL

À découvrir dans cette partie...

2.1 Connaître le matériel dont on dispose

Au fil des années, les ordinateurs sont devenus des machines très complexes, composées de dizaines d'éléments qui interagissent plus ou moins joyeusement. Découvrons les outils qui permettent d'explorer son ordinateur et savoir, enfin, de quoi il est constitué. p. 38

2.2 Mettre en place une connexion Bluetooth

Technologie assez récente, le Bluetooth a surtout été vu dans le cadre d'un environnement utilisateur graphique. Mais il est tout à fait possible d'exploiter cette technologie en ligne de commandes, avec différents outils parfaitement adaptés. p. 46

2 MATÉRIEL

CONNAÎTRE LE MATÉRIEL DONT ON DISPOSE

Mais au fait, j'ai quoi comme matériel ? Vous êtes-vous déjà posé la question ? Que ce soit pour résoudre des problèmes, pour savoir ce qu'on a acheté, ou pour être sûr de ce qu'on a loué (dans le cadre d'un serveur dédié par exemple), différentes commandes permettent d'obtenir ces informations.

1. LE CONTEXTE

Avant de nous intéresser au matériel lui-même, évoquons d'abord les commandes permettant d'obtenir des informations sur le système en lui-même, l'environnement logiciel dans lequel on évolue. Commençons par une commande toute simple, permettant de connaître la version du noyau Linux utilisé :

Terminal

```
$ uname -r
3.11.0-18-generic
```

On peut également obtenir plus d'informations (nom de la machine, type d'architecture...) avec l'argument **-a** :

Terminal

```
$ uname -a
Linux amaretto 3.11.0-18-generic #32-Ubuntu SMP Tue Feb 18 21:11:14 UTC
2014 x86_64 x86_64 x86_64 GNU/Linux
```

Pour obtenir des informations sur la distribution elle-même, ainsi que sa version, c'est la commande **lsb_release** qu'il faut utiliser (LSB signifie *Linux Standard Base*) :

Terminal

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 13.10
Release:        13.10
Codename:       saucy
```

Rapprochons-nous maintenant du matériel, mais tout en restant du côté du système : la commande **lsmod** permet de visualiser la liste des modules du noyau (entre autres, les pilotes de périphériques) qui sont chargés :

Terminal

```
$ lsmod
Module                  Size  Used by
usb_storage            62062  0
[...]
libahci                32009  1 ahci
e1000e                254604  0
```

On peut bien sûr utiliser **grep** pour filtrer le retour de la commande, par exemple :

Terminal

```
$ lsmod | grep i915
i915                  661339  5
i2c_algo_bit           13413  1 i915
drm_kms_helper         52710  1 i915
drm                   297056  6 i915,drm_kms_helper
video                  19318  1 i915
```

Lorsque l'on s'intéresse à un module en particulier, on peut obtenir des informations détaillées sur celui-ci avec la commande **modinfo** :

Terminal

```
$ modinfo e1000e
filename:       /lib/modules/3.11.0-18-generic/kernel/drivers/
net/ethernet/intel/e1000e/e1000e.ko
version:        2.3.2-k
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
[...]
```

2. IDENTIFIER LE MATÉRIEL

2.1 Le processeur

Les informations du processeur sont directement desservies par le noyau Linux, on peut l'interroger avec la simple commande **cat**, aucune commande spéciale n'est nécessaire ; ces informations sont dans le pseudo-fichier **/proc/cpuinfo** :

Terminal

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz
stepping        : 9
microcode       : 0x17
cpu MHz         : 1200.000
cache size      : 3072 KB
[...]
```

Ce pseudo-fichier présente un bloc de ce type pour chaque cœur de processeur détecté sur la machine (un processeur quadricœur sera vu comme quatre processeurs séparés). Les informations présentées sont assez nombreuses : nom du fabricant, nom du modèle, fréquence actuelle, fonctionnalités supportées, présence ou non d'une FPU (unité de calcul en virgule flottante) et de différents autres éléments.

2.2 Les disques

La commande dédiée à la gestion de bas niveau des disques durs est **hdparm**. Cet utilitaire nécessite les droits de super-utilisateur (*root*) afin de fonctionner. Il accepte différents paramètres, à commencer par **-i** (information) ; il faut également lui fournir le chemin vers le périphérique du disque dur : pour le premier disque du système, ce sera **/dev/sda** :

Terminal

```
$ sudo hdparm -i /dev/sda

/dev/sda:

 Model=HGST HTS725050A7E630, FwRev=GH2ZB550, SerialNo=TF755AWHJ09YUM
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=4
BuffType=DualPortCache, BuffSize=unknown, MaxMultSect=16, MultSect=16
```

Définition

Le standard ATA (*Advanced Technology Attachment*), apparu en 1994, décrit une interface de connexion de périphériques de stockage sur PC. Ce standard est plus connu sous le nom IDE (*Integrated Drive Electronics*). Les disques sont reliés au contrôleur (généralement la carte mère) via une nappe composée de 40 ou 80 fils parallèles.

Le standard Serial ATA (ou SATA) est apparu en 2003 ; comme son nom l'indique, il se base sur une communication en série et il est conçu pour supporter de hautes fréquences (en parallèle, on peut facilement subir des désynchronisations à partir de certains fréquences). Depuis l'apparition du SATA, le standard ATA est appelé PATA (*Parallel ATA*), afin de bien différencier les deux normes.

Terminal

```
CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=976773168
IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
PIO modes:  pio0 pio1 pio2 pio3 pio4
DMA modes:  mdma0 mdma1 mdma2
UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 *udma6
AdvancedPM=yes: mode=0x80 (128) WriteCache=enabled
Drive conforms to: unknown:  ATA/ATAPI-2,3,4,5,6,7

* signifies the current active mode
```

La première ligne nous renseigne sur la marque et le modèle du disque ; les autres nous informent sur ses particularités matérielles et sur les fonctionnalités qu'il supporte. On peut également utiliser l'option **-I** afin d'obtenir des informations encore plus détaillées.

L'option **-t**, quant à elle, permet de tester la vitesse de lecture du disque dur (les premiers secteurs sont testés pendant 3 secondes) ; afin d'obtenir des informations fiables, il est préférable d'exécuter la commande trois fois d'affilée :

Terminal

```
$ sudo hdparm -t /dev/sda

/dev/sda:
 Timing buffered disk reads: 384 MB in  3.01 seconds = 127.76 MB/sec
```

Les disques durs sont composés de partitions ; pour visualiser les partitions existantes sur un disque dur, c'est la commande **parted** qui est utilisée – son argument **-l** permet de lister toutes les partitions de l'ensemble des disques :

Terminal

```
$ sudo parted -l
Modèle: ATA HGST HTS725050A7 (scsi)
Disque /dev/sda : 500GB
Taille des secteurs (logiques/physiques): 512B/4096B
Table de partitions : gpt

Numéro  Début   Fin     Taille  Système de fichiers  Nom  Fanions
 1      1049kB  99,6MB  98,6MB  fat32                     démarrage
 2      99,6MB  30,8GB  30,7GB  ext4                      msftdata
 3      30,8GB  32,9GB  2048MB  linux-swap(v1)
 4      32,9GB  500GB   467GB   ext4                      msftdata
```

Si **parted** n'est pas installé, on peut également utiliser la commande **fdisk -l**, qui présente ces informations sous une forme un peu plus difficile à comprendre…

On peut également vouloir savoir où et comment sont montées les différentes partitions sur le système. Pour ce faire, la commande adéquate est **mount**. Notons que celle-ci montre tous les volumes montés, que ce soit des partitions de disques durs ou des volumes virtuels (communication avec le noyau par **/proc** ou **/sys** par exemple).

Terminal

```
$ mount
/dev/sda2 on / type ext4 (rw,errors=remount-ro)
[...]
/dev/sda1 on /boot/efi type vfat (rw)
/dev/sda4 on /home type ext4 (rw)
[...]
```

Vous pouvez également vouloir contrôler de temps en temps l'espace disponible. Cela se fait avec la commande **df**, à laquelle on ajoute l'option **-h** afin d'afficher des informations « lisibles par un **h**umain » :

Terminal

```
$ df -h
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sda2           29G    9,2G   18G  35% /
[...]
/dev/sda1           93M    3,3M   90M   4% /boot/efi
/dev/sda4          429G    397G   32G  93% /home
```

Cette commande retourne les informations d'espace disque sur tous les volumes actuellement montés, dont les partitions des disques durs.

2.3 Les cartes et adaptateurs

Pour identifier les cartes et adaptateurs reliés aux bus PCI, AGP ou PCI Express, on utilise la commande **lspci**. Cela concerne tous les adaptateurs intégrés à la carte mère ou connectés à l'un de ces bus : carte son, carte réseau, carte graphique, etc.

lspci utilisé seul retournera un résumé de l'ensemble des périphériques, tandis que **lspci -v** et **lspci -vv** présentent des informations de plus en plus détaillées (fréquence du bus, zones mémoire occupées, etc.). L'option **-n** retourne les identifiants numériques des périphériques, l'option **-t** les présente sous une vue arborescente. On peut également filtrer les périphériques à afficher avec les options **-s** (adresse de périphérique) ou **-d** (identifiant).

Terminal

```
$ lspci
00:00.0 Host bridge: Intel Corporation 3rd Gen Core processor DRAM Controller (rev 09)
00:02.0 VGA compatible controller: Intel Corporation 3rd Gen Core processor Graphics
Controller (rev 09)
[...]
02:00.0 System peripheral: Ricoh Co Ltd PCIe SDXC/MMC Host Controller (rev 07)
03:00.0 Network controller: Intel Corporation Centrino Wireless-N 2200 (rev c4)
```

2.4 Les périphériques USB

La liste des périphériques USB (clés, disques, claviers, souris, webcams, etc.) est obtenue avec la commande **lsusb** :

Terminal

```
$ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[...]
Bus 003 Device 002: ID 046d:c52f Logitech, Inc. Unifying Receiver
```

Comme pour de nombreuses autres commandes, on obtiendra des informations plus détaillées avec l'option **-v** ; les options **-s** et **-d** permettent de filtrer les périphériques à afficher, respectivement par bus ou par identifiant numérique.

Certaines informations détaillées ne sont disponibles que pour *root*, auquel cas il faut exécuter la commande avec **sudo**...

2.5 Le « couteau suisse »

Si vous ne deviez retenir qu'une seule commande dans le cadre de l'identification du matériel, ce serait **lshw** (list hardware) ; ce programme réunit de très nombreuses informations, qui pourront recouper,

À savoir

Afin de présenter des noms humainement compréhensibles au lieu d'identifiants numériques, lspci et lusb utilisent des bases de données d'identifiants connus. Celles-ci sont stockées respectivement dans les fichiers /usr/share/misc/pci.ids et /var/lib/usbutils/usb.ids (sur Debian et Ubuntu). On peut mettre ces bases à jour avec les commandes sudo update-pciids et sudo update-usbids.

remplacer ou compléter celles des commandes que nous avons vues précédemment. Cette commande n'est toutefois pas disponible sur tous les systèmes, ce n'est pas une commande « historique » ; sa toute première version date de 2003 : parfois installée par défaut, elle sera souvent retrouvée dans les dépôts de la distribution, mais à installer avant de pouvoir l'utiliser.

Il est par ailleurs préférable de l'exécuter en tant que *root* (sous Ubuntu, avec la commande **sudo**), afin d'obtenir toutes les informations disponibles.

Dans la mesure où cette commande retourne énormément de données, il est préférable de la rediriger vers **less**, afin de pouvoir naviguer dans tout ce qui est retourné :

Terminal

```
$ sudo lshw | less
[...]
     *-cpu
          description: CPU
          product: Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz
          vendor: Intel Corp.
[...]
     *-memory
          description: System Memory
          physical id: 7
          slot: System board or motherboard
          size: 16GiB
[...]
```

On peut également obtenir ces données sous une forme plus agréable à consulter, dans un fichier HTML, avec la commande suivante :

Terminal

```
$ sudo lshw -html > mon_materiel.html
```

Mentionnons aussi la commande **dmidecode**, qui sert à lire le contenu de la table DMI (*Desktop Management Interface*) de l'ordinateur : état actuel de la machine, quantité maximale de mémoire vive, vitesse maximale du processeur, etc. La table DMI correspond aux informations telles qu'elles sont détectées par le BIOS et la carte mère. Cela étant dit, les informations de **lshw** sont également tirées de la table DMI : en réalité, cette commande est suffisante pour explorer le matériel.

3. SURVEILLER LE MATÉRIEL

3.1 Disques durs

SMART (*Self-Monitoring Analysis and Reporting Technology*) est un système de surveillance des disques durs. Il permet d'anticiper les défaillances d'un disque grâce à différents indicateurs ; cette technologie doit être supportée par le disque dur concerné : c'est le cas pour la plupart des disques actuels.

Sur de nombreuses distributions Linux (dont Debian et Ubuntu), le support de SMART est assuré par le paquet nommé **smartmontools**.

Terminal

```
$ sudo apt-get install smartmontools
```

Dans certains cas, avant d'analyser un disque, il faut y activer SMART :

Terminal

```
$ sudo smartctl -s on /dev/sda
```

Pour obtenir les données SMART sur un disque dur (par exemple **/dev/sda**), la commande **smartctl** doit être utilisée avec l'option **-a** :

Terminal

```
$ sudo smartctl -a /dev/sda
smartctl 6.2 2013-04-20 r3812 [x86_64-linux-3.11.0-18-generic] (local build)
Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Device Model:     HGST HTS725050A7E630
Serial Number:    TF755AWHJ09YUM
[...]
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x000b   100   100   062    Pre-fail  Always       -       0
[...]
193 Load_Cycle_Count        0x0012   098   098   000    Old_age   Always       -       23719
194 Temperature_Celsius     0x0002   171   171   000    Old_age   Always       -       35 (Min/Max 11/46)
[...]
```

C'est la section « *Vendor Specific SMART Attributes with Thresholds* » qui est la plus intéressante : on y retrouve différentes informations sur le fonctionnement du disque dur, comme la température du disque (**Temperature_Celcius**), la durée totale du fonctionnement (**Power_On_Hours**), le nombre d'erreurs de différents types...

Afin de surveiller uniquement la température des disques, on peut utiliser la commande **hddtemp** (fournie par le paquet **hddtemp**) ; celle-ci s'appuie sur les données de SMART : on obtiendra alors la même information qu'avec **smartctl**, mais présentée de manière plus concise :

Terminal

```
$ sudo hddtemp /dev/sda
/dev/sda: HGST HTS725050A7E630: 25°C
```

On peut alors facilement intégrer cette information dans un programme tiers de surveillance.

3.2 Capteurs divers

Par ailleurs, saviez-vous que votre PC est doté de nombreux capteurs, disséminés un peu partout ? Tous ces capteurs sont reliés ensemble sur un bus appelé I2C (*Inter-Integrated Circuit*). Capteurs de température, de tension... Toutes ces données peuvent être récupérées ne serait-ce que pour vous assurer que tout va bien dans les entrailles de la machine. Ces informations peuvent être récupérées grâce aux outils développés dans le cadre du projet Lm-sensors (distribué par Debian et Ubuntu dans le paquet **lm-sensors**, tout simplement).

Tout d'abord, il est nécessaire de charger les modules permettant d'activer le support du bus I2C et des différents capteurs. Afin de les identifier, on peut utiliser la commande **sensors-detect** ; celle-ci pose une série de questions auxquelles il faudra répondre par l'affirmative pour tester les différents modules. Enfin, la commande indique une liste de modules qu'il faut charger afin de supporter votre matériel.

Terminal

```
$ sudo sensors-detect
[...]
To load everything that is needed, add this to /etc/modules:
#----cut here----
# Chip drivers
coretemp
#----cut here----
```

Après cela, cette commande propose de mettre en place ces lignes dans le fichier **/etc/modules** automatiquement. Une fois ceci fait, il faut lancer le processus de surveillance :

Terminal
```
$ sudo service kmod start
```

À présent, la commande **sensors** peut être utilisée pour consulter ces indicateurs :

Terminal
```
$ sensors
acpitz-virtual-0
Adapter: Virtual device
temp1:        +41.0°C  (crit = +103.0°C)

thinkpad-isa-0000
Adapter: ISA adapter
fan1:           0 RPM

coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +42.0°C  (high = +87.0°C, crit = +105.0°C)
Core 0:         +42.0°C  (high = +87.0°C, crit = +105.0°C)
Core 1:         +42.0°C  (high = +87.0°C, crit = +105.0°C)

pkg-temp-0-virtual-0
Adapter: Virtual device
temp1:        +42.0°C
```

3.3 Messages du noyau

Bien que généralement on n'en a pas conscience, le noyau Linux émet beaucoup de messages, à chaque événement qu'il rencontre. Pour lire ces messages, on utilise la commande **dmesg**. On peut filtrer son retour avec **grep** par exemple :

Terminal
```
$ dmesg | grep -i usb
```

Au moment où on rencontre un problème, on peut également visualiser uniquement les dernières lignes (30 dans cet exemple), avec la commande suivante :

Terminal
```
$ dmesg | tail -n 30
```

Cette dernière est très utile pour commencer à chercher d'où vient une panne.

CONCLUSION

Si vous avez testé toutes les commandes évoquées ci-dessus, votre matériel ne devrait plus avoir de secret pour vous. Vous serez alors plus à même de rechercher des informations pour améliorer votre configuration, activer le support d'un périphérique en trouvant le bon pilote, ou encore comparer les performances de certains matériels et, bien sûr, résoudre d'éventuels problèmes que vous pourrez rencontrer...

Pour aller plus loin, n'hésitez pas à consulter les pages de manuel de toutes ces commandes : nous avons abordé ici leurs principales options, mais d'autres pourront peut-être vous aider à répondre à votre problématique particulière... ■

METTRE EN PLACE UNE CONNEXION BLUETOOTH

Le Bluetooth est une technologie relativement récente (version 1.0 en 1999) et on a l'habitude de l'utiliser avec des interfaces graphiques. Mais on peut tout à fait utiliser des périphériques Bluetooth en ligne de commandes !

Le Bluetooth est une technologie plus simple à mettre en œuvre qu'il n'y paraît ; de plus, c'est maintenant un mode de communication presque universel pour de la connexion sans fil à courte portée. En effet, la seule autre technologie répandue, le Wi-Fi, est lourde à mettre en place dans des montages électroniques par exemple ; des petits modules existent, mais il faut dans ce cas gérer toute une pile IP ; en cela, le Bluetooth est plus simple, ses protocoles étant adaptés à des communications légères ; du point de vue des montages électroniques, il existe des petits modules à connexion série capables de communiquer avec n'importe quel ordinateur sachant prendre en charge les fonctions rfcomm/SSP (tous, donc).

Cependant, avant de communiquer en Bluetooth (que ce soit dans le cadre d'un montage électronique ou pour l'utilisation d'un périphérique Bluetooth quelconque), il faut **associer** l'ordinateur et le périphérique (*pairing*), cela implique la plupart du temps une **authentification**. Lorsqu'on utilise un environnement graphique, cette authentification prend généralement la forme d'une fenêtre « pop-up » demandant un code PIN. De la même manière, la recherche et l'association de périphériques se font la plupart du temps par une interface graphique intégrée à l'environnement de bureau.

Pendant longtemps, associer un périphérique à un code PIN relevait d'un bricolage assez sale, consistant à modifier manuellement des fichiers de configuration, pour y insérer cette information qui est, par essence, dynamique. On n'a plus besoin, aujourd'hui, de faire de tels bricolages : on peut maintenant utiliser la commande **bluetooth-agent**, qui a pour objectif de gérer les codes PIN ; il communique avec le démon Bluetooth **bluetoothd** par l'intermédiaire de D-Bus. Ce dernier est une infrastructure qui permet à différents logiciels de communiquer entre eux, centralisant ce qui était, il y a de longues années, un ensemble de flux disparates.

D-Bus est particulièrement bien adapté aux environnements de bureau, où l'utilisateur s'attend à ce que tous les logiciels interagissent afin de lui faciliter la vie. Comme n'importe quelle surcouche, D-Bus est une horreur à gérer en ligne de commandes (on l'utilise avec la commande **dbus-send**, n'hésitez pas à vous renseigner à son sujet si cela vous intéresse !). Heureusement, tout cela se fait de manière transparente grâce à l'agent sus-cité. Notons que cet agent était à l'origine développé à des fins de tests, il a fallu longtemps avant qu'il n'obtienne le statut d'utilitaire à part entière. Si vous cherchez sur le Web des informations sur le Bluetooth en ligne de commandes sous GNU/Linux et que vous ne trouvez pas de mention de **bluetooth-agent**, il y a de fortes chances que les explications données ne soient pas à jour.

1. DÉCOUVRIR LES PÉRIPHÉRIQUES

La première étape dans l'utilisation du Bluetooth consiste généralement à découvrir les périphériques à portée. Cela se fait avec l'option **scan** de l'outil générique **hcitool** :

Terminal

```
$ hcitool scan
Scanning ...
        BC:20:A4:23:8E:0D       GT-N7100
        00:12:47:01:23:45       Samsung Galaxy S
```

Notons qu'il n'est pas nécessaire d'avoir les privilèges de super-utilisateur pour cette découverte, mais d'autres actions en ont besoin. Par exemple, pour obtenir plus d'informations sur un périphérique en particulier, il faut utiliser **sudo** :

Terminal

```
$ hcitool info 00:12:47:01:23:45
Requesting information ...
Can't create connection: Operation not permitted
$ sudo hcitool info 00:12:47:01:23:45
Requesting information ...
        BD Address:  00:12:47:01:23:45
        Device Name: Samsung Galaxy S
        LMP Version: 2.1 (0x4) LMP Subversion: 0x4217
[...]
```

2. IDENTIFIER LES SERVICES PROPOSÉS

Le protocole Bluetooth proposes plusieurs méthodes, plusieurs services. On trouvera par exemple RFCOMM, qui permet de simuler un port série (on trouve d'ailleurs des adaptateurs série/bluetooth peu onéreux, permettant de rendre sans-fil n'importe quelle communication sur port série) ; la commande **rfcomm** utilise ce service pour créer un pseudo-port série **/dev/rfcomm0**.

Le protocole OBEX, quant à lui, permet de transférer des fichiers d'un appareil à un autre.

Pour découvrir tous les services proposés par les périphériques à proximité, il faut utiliser la commande **sdptool browse** ; sans argument, elle montre les services de tous les périphériques. On peut également lui préciser l'identifiant auprès duquel demander la liste des services. Cette commande utilise le protocole SDP (*Service Discovery Protocol*), qui est lui-même un service des périphériques Bluetooth.

Terminal

```
$ sdptool browse 00:12:47:01:23:45
Browsing 00:12:47:01:23:45 ...
[...]
Service Name: OBEX File Transfer
Service RecHandle: 0x1000a
Service Class ID List:
  "OBEX File Transfer" (0x1106)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 20
  "OBEX" (0x0008)
Profile Descriptor List:
  "OBEX File Transfer" (0x1106)
    Version: 0x0101
```

Parmi tous les services proposés par ce téléphone, on trouve bien un service *OBEX File Transfer* : on peut donc effectuer des transferts de fichiers.

3. APPARIER UN PÉRIPHÉRIQUE

Avant de se connecter à un périphérique, il faut l'associer, autrement dit apparier l'ordinateur et le périphérique. C'est là qu'entre en jeu **bluetooth-agent**. On peut l'utiliser de deux manières :

⇨ Soit en donnant un code PIN par défaut, avec la syntaxe suivante :

Terminal

```
$ bluetooth-agent 0000
```

⇨ Soit en précisant le lien entre l'identifiant du périphérique Bluetooth et un code PIN :

Terminal

```
$ bluetooth-agent 0000 12:34:56:78:90:12
```

Dans le premier cas, **bluetooth-agent** reste actif (il ne rend pas la main) jusqu'à ce qu'un code PIN soit demandé par le démon **bluetoothd** : il répond alors avec le code que vous avez précisé par avant. La suite des opérations doit être effectuée dans un autre terminal.

Dans le second cas, **bluetooth-agent** transmet l'association entre le code PIN et l'identifiant du périphérique à **bluetoothd**, qui va directement essayer de s'associer au périphérique : le même code PIN est alors demandé sur notre téléphone. Une fois renseigné, **bluetooth-agent** rend la main :

À savoir

Les services proposés par un périphérique Bluetooth peuvent en réalité être très nombreux. Le téléphone pris en exemple ici propose pas moins de 10 services, dont l'accès aux SMS, aux e-mails, le contrôle de lecteurs multimédias...

Terminal

```
$ bluetooth-agent 1234 00:12:47:01:23:45
Pincode request for device /org/bluez/943/hci1/dev_00_12_47_01_23_45
Agent has been released
```

L'ordinateur apparaît alors dans les périphériques associés du téléphone (dans l'écran de paramétrage du Bluetooth). Le message « *Agent has been released* » apparaît une fois que le code PIN a été tapé sur le téléphone. Les informations liées aux périphériques Bluetooth (code PIN, etc.) sont stockées par **bluetoothd** dans le répertoire **/var/lib/bluetooth**, un sous-répertoire par périphérique.

4. SE CONNECTER AU PÉRIPHÉRIQUE

Nous allons transférer un fichier de l'ordinateur vers le téléphone. Pour cela, on utilisera la commande **obexftp** – il faut installer le paquet du même nom.

Pour envoyer un fichier à un appareil en Bluetooth, il faut utiliser la syntaxe suivante : **obexftp -b <identifiant> -p <fichier>**.

Lors d'un tel transfert, le téléphone demande une autorisation (celle-ci peut être accordée définitivement, auquel cas, si un transfert a déjà eu lieu, la question n'est pas posée).

Ce qui donne, dans notre cas pratique :

Terminal

```
$ obexftp -b 00:12:47:01:23:45 -p awk_principes.pdf
Browsing 00:12:47:01:23:45 ...
Connecting...done
Tried to connect for 579ms
Sending "awk_principes.pdf"....done
Disconnecting...done
```

Et voilà, le fichier est maintenant présent sur le téléphone !

On peut également lister les fichiers sur le téléphone avec l'option **-l** :

Terminal

```
$ obexftp -b 00:12:47:01:23:45 -l /
Browsing 00:12:47:01:23:45 ...
Connecting...done
Tried to connect for 775ms
Receiving "/"... Sending ""...done
<?xml version="1.0"?>
<!DOCTYPE folder-listing SYSTEM "obex-folder-listing.dtd">
<folder-listing version="1.0">
<folder name="external_sd" size="32768" user-perm="RW" modified="20131229T115000Z"/>
[...]
<file name="awk_principes.pdf" size="340716" user-perm="RW" modified="20140326T102900Z"/>
</folder-listing>
done
Disconnecting...done
```

L'option **-g**, quant à elle, permet de télécharger un fichier du téléphone. Il en existe également d'autres, n'hésitez pas à consulter la page de manuel d'**obexftp**.

Sous Linux, on trouve aussi le logiciel **obexfs**, qui permet de « monter » le stockage distant dans l'arborescence UNIX classique ; il utilise FUSE, qui permet de monter comme systèmes de fichiers de nombreux espaces de stockage qui ne sont pas initialement prévus pour cela.

CONCLUSION

Nous avons décrit ici une utilisation du Bluetooth. On a également évoqué la simulation d'un port série avec RFCOMM, on peut aussi penser aux usages liés à l'audio (oreillette Bluetooth, haut-parleurs déportés, etc.). Chacun trouvera à ce protocole ses avantages et l'utilisera comme il le souhaitera : c'est un domaine relativement vaste, qu'on pourrait passer des journées entières à approfondir. ■

3

3 SYSTÈME

À découvrir dans cette partie...

IDENTIFIER LES CAUSES D'UN RALENTISSEMENT

Une distribution Linux, ce n'est pas magique : comme n'importe quel système, ça peut « ramer », pour différentes raisons : système surchargé, processus qui s'emballe, mémoire vive insuffisante... Que peut-on y faire ?

Dans un système informatique, de nombreux processus fonctionnent en parallèle ; ils correspondent aux différents logiciels en fonctionnement. Ces processus consomment différentes ressources, notamment du processeur et de la mémoire vive. Il peut arriver qu'un processus devienne trop gourmand, pour des raisons légitimes ou non. De plus, la quantité de mémoire est limitée et certains programmes sont particulièrement voraces dans ce domaine.

1. ÉTAT DU SYSTÈME

Dans un premier temps, étudions les commandes permettant d'analyser l'utilisation des différentes ressources de l'ordinateur...

1.1 Le processeur

Sur un système UNIX, on ne parle généralement pas de pourcentage d'utilisation du processeur, mais de **charge**. La charge d'un processeur, c'est la moyenne du nombre de processus utilisant ou attendant le processeur, sur une durée donnée.

Une charge de 1 correspond à l'utilisation d'un processeur à 100 % pendant la durée concernée. Une charge inférieure à 1 indique que la machine est sous-utilisée. Enfin, une charge supérieure à 1 peut indiquer qu'il y a des processus en attente. Mais attention : un processus en attente, ce n'est pas nécessairement un problème ! On peut rencontrer des serveurs (web, par exemple) à fort trafic ayant une charge de plusieurs centaines, mais qui répondent de manière suffisamment réactive. De plus, ce compte se fait par processeur (par cœur) : par exemple, sur une machine quadri-processeur ou quadri-cœur, la machine n'est pas surchargée tant que la charge ne dépasse pas 4.

Pour lire la charge du processeur, on utilise la commande **uptime** :

Terminal

```
$ uptime
 22:10:45 up 2 days,  3:31,  8 users,  load average: 0,12, 0,15, 0,14
```

Cette commande retourne l'heure courante, la durée de fonctionnement depuis le dernier allumage, le nombre de sessions utilisateurs ouvertes et la charge moyenne du système (3 valeurs).

Les trois dernières valeurs correspondent à la charge moyenne sur les 1, 5 et 15 dernières minutes. Si le premier nombre (moyenne sur 1 minute) est supérieur aux autres, cela veut dire que l'activité est importante depuis peu de temps. Si, au contraire, le dernier nombre (moyenne sur 15 minutes) est supérieur aux autres, cela veut dire que l'activité est en train de se réduire.

1.2 La mémoire

La mémoire vive peut s'avérer insuffisante pour certains usages, d'autant plus que les nouvelles versions des logiciels sont généralement de plus en plus gourmandes. Lorsque la mémoire vive est pleine, le système utilise l'espace de swap (lorsqu'il y en a), rendant ainsi l'utilisation de l'ordinateur pénible.

De nos jours, les barrettes de mémoire vive ne sont pas très onéreuses : le swap est surtout là pour qu'on ait le temps de se rendre compte qu'il faut ajouter de la mémoire vive à l'ordinateur !

Pour consulter l'espace disponible en mémoire vive et sur le swap, on utilise la commande **free** ; l'argument **-m** est utilisé pour afficher les valeurs en méga-octets, rendant plus simple la lecture des données :

Terminal

```
$ free -m
             total       used       free     shared    buffers     cached
Mem:         15867       7389       8477          0        473       3532
-/+ buffers/cache:       3384      12483
Swap:         1952          0       1952
```

Définition

Le swap est un espace mémoire sur le disque dur ; son objectif est de pallier le manque de mémoire vive. Plutôt que de faire planter un logiciel qui serait trop gourmand, l'ordinateur utilise alors le swap : le système est très ralenti (le disque dur étant des centaines de fois plus lent que la mémoire vive), mais les logiciels ne plantent pas. Le swap est également utilisé par l'hibernation (mise en veille prolongée) : les données de la mémoire vive et l'état du système y sont alors enregistrés pour être récupérés après l'allumage de l'ordinateur.

À première vue, on constate que cet ordinateur a environ 7,4 Go de mémoire vive utilisée et 8,5 Go libres, pour un total de presque 16 Go. On remarque également que le swap, de 2 Go environ, n'est pas utilisé.

Mais ce n'est pas tout ! Les notions de **buffer** et de **cache** sont très importantes. Un buffer est un espace de mémoire de transit partagé entre plusieurs éléments, par exemple un processus et le noyau pour un transfert de données, afin de permettre au processus de ne pas attendre un matériel lent : par exemple, le noyau se charge de l'écriture sur le disque dur, tandis que le processus peut continuer à fonctionner comme si les données étaient déjà écrites. La mémoire cache, quant à elle, est un espace qui a été utilisé par le passé et qui peut éventuellement être réutilisé tel quel. Par exemple, si vous fermez LibreOffice et le rouvrez quelques secondes après, son lancement est plus rapide qu'à froid : une partie des données nécessaires à son fonctionnement sont déjà dans la mémoire vive, directement utilisables.

Lorsque les processus ont besoin de beaucoup de mémoire pour fonctionner, le noyau réduit automatiquement l'espace alloué aux buffers et (surtout) au cache : ces données ne sont absolument pas nécessaires pour le fonctionnement du système, l'espace est conservé « au cas où », mais peut être effacé à tout moment. C'est pourquoi la mémoire « réellement utilisée » et l'espace utilisable sont indiqués dans la ligne **`-/+ buffers/cache`** : ici, 3,4 Go de RAM sont réellement utilisés et 12,5 Go sont disponibles.

La colonne **`shared`**, quant à elle, est obsolète ; elle est conservée uniquement pour des raisons de compatibilité, par exemple pour de vieux scripts qui utiliseraient cette commande.

Lorsque l'on souhaite des informations plus détaillées sur la mémoire, on utilise la commande **`vmstat`**. Celle-ci permet de visualiser en temps réel les valeurs liées à la mémoire virtuelle (vm, *virtual memory*) ; elles ne sont pas limitées à l'utilisation de la mémoire vive et du swap. Cette commande accepte deux arguments : le premier est le nombre de secondes entre deux mises à jour de données, le second est le nombre de fois où l'interrogation doit être effectuée. En l'absence de ces arguments, les données ne sont récupérées qu'une fois.

Terminal

```
$ vmstat 2 3
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 1  0      0 8580228 485312 3652184    0    0    23    69  259  201 13  3 83  0
[...]
```

Les colonnes indiquent les informations suivantes :

- **`r`** : nombre de processus en attente du processeur ;
- **`b`** : nombre de processus en sommeil non-interruptible ;
- **`swpd`** : quantité de swap utilisée ;
- **`free`** : quantité de mémoire libre ;
- **`buff`** : quantité de mémoire attribuée aux buffers ;
- **`cache`** : quantité de mémoire cache ;
- **`si`** : quantité de mémoire lue du swap par seconde (*swap in*) ;
- **`so`** : quantité de mémoire écrite dans le swap par seconde (*swap out*) ;
- **`bi`** : quantité de mémoire lue d'un périphérique de type disque dur, par seconde (*block in*) ;
- **`bo`** : quantité de mémoire écrite sur un périphérique de type disque dur, par seconde (*block out*) ;
- **`in`** : nombre d'interruptions système par seconde ;
- **`cs`** : nombre de changements de contexte par seconde ;

- **us** : pourcentage d'utilisation du processeur par les processus en espace utilisateur ;
- **sy** : pourcentage d'utilisation du processeur par le noyau ;
- **id** : pourcentage de non-utilisation du processeur ;
- **wa** : pourcentage de temps processeur passé à attendre des entrées/sorties.

Ceux qui veulent des informations encore plus détaillées peuvent consulter le contenu du pseudo-fichier **/proc/meminfo**, qui contient plus de 40 lignes ; si vous êtes intéressé par ces informations, n'hésitez pas à chercher leur signification sur Internet !

2. IDENTIFIER LE COUPABLE

Un ralentissement vient quasiment toujours d'un processus qui « s'emballe », différents outils peuvent être utilisés pour analyser les processus ; les deux principaux sont **ps** et **top**.

2.1 À un instant t : ps

Commençons par la commande **ps**. Lorsque l'on exécute cette commande sans argument, elle retourne la liste des processus associés au terminal courant, avec un niveau de détail limité. On peut alors utiliser différents arguments afin de changer le comportement de cette commande, notamment :

- **a** : affiche les processus de tous les utilisateurs ;
- **u** : affiche des informations détaillées ;
- **x** : affiche tous les processus (non limité aux processus liés à un terminal) ;
- **w** : affichage large – utilisé deux fois, l'affichage n'est pas limité en largeur, les commandes sont alors affichées dans leur intégralité.

Précisons quelque chose : on utilise ici la syntaxe BSD, sans tiret avant les arguments. Les mêmes options avec la syntaxe GNU (précédées par un tiret) ont une signification différente.

Terminal

```
$ ps auxww
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
[...]
root      1162  1.0  1.2 503852 200728 tty7    Ssl+ mars24  42:14 /usr/bin/X -core :0 -auth
/var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
1000      1897  0.0  0.0   4204    88 ?        S    mars24   0:00 /usr/lib/kde4/libexec/start_
kdeinit +kcminit_startup
1000      1898  0.0  0.1 347056 22876 ?        Ss   mars24   0:02 kdeinit4: kdeinit4 Running...
[...]
```

ps auxww retourne 11 colonnes :

- **USER** : l'utilisateur propriétaire du processus ;
- **PID** : l'identifiant unique du processus ;
- **%CPU** : le pourcentage actuel d'utilisation du processeur par le processus ;
- **%MEM** : le pourcentage actuel d'utilisation de la mémoire par le processus ;
- **VSZ** : la taille de la mémoire virtuelle allouée au processus ;
- **RSS** : la mémoire physique réellement utilisée par le processus ;
- **TTY** : le terminal auquel le processus est rattaché, beaucoup de processus n'étant liés à aucun terminal ;
- **STAT** : l'état actuel du processus ;
- **START** : le moment auquel le processus a démarré (une heure si le processus a démarré dans la journée, sinon une date) ;
- **TIME** : le temps de processeur utilisé par le processus depuis son lancement ;
- **COMMAND** : la commande utilisée pour démarrer le processus.

Pour diagnostiquer les processus, il faut s'intéresser aux colonnes suivantes :

- **%CPU** : dans notre cas, il y a par exemple le serveur graphique X.Org (commande **/usr/bin/X**), qui utilise 1,0 % du processeur – lorsque l'utilisation du processeur avoisine les 100 % pendant longtemps, cela peut être un signe de problème ;
- **%MEM** : ici, X.Org utilise 1,2 % de la mémoire vive – si l'utilisation de la mémoire était excessive, cela serait probablement le signe d'un problème, sauf si on s'attend à ce que le programme concerné consomme beaucoup de mémoire bien sûr (traitement vidéo ou photo par exemple) ;
- **TIME** : le « temps de processeur » représente la durée totale pendant laquelle le processus a utilisé un processeur – en général, même pour un processus très gourmand, cette valeur reste relativement basse – ici, le serveur X a un temps de processeur cumulé de 42 minutes et 14 secondes, depuis son exécution le 24 mars (la commande **ps** a été lancée le 27 mars) : 42 minutes sur 3 jours, ce n'est pas excessif pour un programme aussi important que le serveur graphique.

La colonne **TIME** est très intéressante lorsqu'un problème est récurrent : si on constate un temps processeur très important (par exemple 15 heures en trois jours), il y a de fortes chances que le processus concerné soit fautif.

2.2 En temps réel : top

La commande **top**, présente sur la majorité des systèmes de type UNIX, permet de visualiser un classement en temps réel des processus selon différents critères ; elle ne prend généralement pas d'argument, elle offre par contre une interface interactive.

Terminal

```
top - 12:01:39 up 2 days, 17:22,  8 users,  load average: 0,13, 0,32, 0,50
Tasks: 255 total,   2 running, 244 sleeping,   0 stopped,   9 zombie
%Cpu(s):  4,1 us,  2,2 sy,  0,0 ni, 93,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:  16248184 total,  7648180 used,  8600004 free,   487720 buffers
KiB Swap:  1999868 total,        0 used,  1999868 free,  3686544 cached
  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2138 sebastie  20   0 6250m 342m  76m S   8,0  2,2  83:24.75 amarok
 1162 root      20   0  450m 174m 145m S   3,3  1,1  42:32.97 Xorg
 1966 sebastie  20   0 3445m 240m  59m S   3,0  1,5  37:03.50 plasma-desktop
[...]
```

Par défaut, cette commande affiche les processus classés par pourcentage d'utilisation du processeur, avec une mise à jour de l'affichage toutes les 3 secondes. Pour modifier l'affichage, il suffit de taper sur une touche du clavier : chaque lettre correspond à une commande transmise à **top** ; par exemple :

- **M** permet de classer les processus par utilisation de la mémoire ;
- **P** permet de classer les processus par utilisation du processeur ;
- **s** permet de changer le délai de mise à jour de l'affichage (on tape ensuite un nombre de secondes, validé par la touche [Entrée]) ;
- **q** permet de quitter **top**.

En complément des informations globales sur le système en haut de l'écran, **top** affiche les colonnes suivantes pour les processus :

- **PID** : numéro unique du processus ;
- **USER** : nom de l'utilisateur propriétaire du processus (tronqué à 8 caractères) ;
- **PR** : priorité du processus ;
- **NI** : « niceness » du processus ;
- **VIRT** : mémoire totale utilisée par le processus ;
- **RES** : mémoire vive utilisée par le processus ;
- **SHR** : mémoire potentiellement partagée avec d'autres processus ;

À savoir

Un processus est, à un moment donné, dans un état donné. Les états possibles sont les suivants :

- **R : en fonctionnement (*running*) ;**
- **S : sommeil interruptible (en attente d'un événement) ;**
- **D : sommeil non-interruptible (en attente d'une entrée/sortie, généralement) ;**
- **T : arrêté (par exemple par la combinaison Ctrl-Z dans un terminal) ;**
- **X : mort (normalement, un processus mort disparaît simplement, cet état ne devrait jamais être rencontré) ;**
- **Z : « zombie », le processus est terminé mais pas nettoyé par son parent – dans ce cas, il faut s'intéresser au parent.**

Les processus dans les états D ou Z ne peuvent pas être arrêtés de force. Si on a des processus « zombie » pour lesquels on ne peut pas gérer le parent (par exemple si le parent est déjà mort, le processus enfant est alors orphelin), il n'y a plus qu'une solution : redémarrer la machine.

- ⇨ **S** : état du processus ;
- ⇨ **%CPU** : pourcentage d'utilisation du processeur par le processus ;
- ⇨ **%MEM** : pourcentage d'utilisation de la mémoire par le processus ;
- ⇨ **TIME+** : temps de processeur consommé par le processus depuis son lancement ;
- ⇨ **COMMAND** : commande utilisée pour démarrer le processus.

3. MANIPULER LES PROCESSUS

Différentes commandes existent pour manipuler les processus, la principale manipulation étant le fait de « tuer » un processus. Ceci peut se faire avec différentes commandes, dont **kill** et **pkill**.

3.1 Tuer selon le PID

Lorsque l'on connaît l'identifiant numérique (le PID) d'un processus, on peut utiliser la commande **kill** pour le tuer. Cette commande prend comme argument au minimum un PID, elle a également différentes options dont **-s** pour transmettre un signal au processus. En effet, en réalité on ne fait pas que « tuer » des processus : on leur envoie des signaux, identifiés par un nom ou un numéro, parmi lesquels :

- ⇨ **1**, **HUP** : *hang-up* – certains processus comprennent ce signal comme une demande de rechargement de la configuration (c'est notamment utile dans le cadre de serveurs, comme Apache) ;
- ⇨ **9**, **KILL** : ce signal n'est pas intercepté par le processus, il indique au système de tuer « sauvagement » le processus, sans lui laisser le temps de fermer des fichiers par exemple ;
- ⇨ **15**, **TERM** : ce signal demande « gentiment » au processus de se terminer proprement, en enregistrant les fichiers ouverts par exemple.

Lorsqu'aucun signal n'est précisé, c'est **TERM** qui est envoyé. La page de manuel de **kill** précise la liste complète des signaux.

Par exemple, pour tuer immédiatement le processus ayant le PID 1337, la commande serait la suivante :

Terminal

```
$ kill -9 1337
```

Bien sûr, si le processus appartient à un autre utilisateur, on ne peut pas le tuer, sauf si on est *root*...

3.2 Tuer par nom de commande

Il arrive qu'on veuille tuer un processus pour lequel on connaît la commande utilisée pour le lancer. Dans ce cas, il faut utiliser **pkill**. Cette commande prend comme argument la chaîne de caractères à rechercher dans les noms de commandes (par défaut, elle ne cherche que dans le nom de l'exécutable). Si plusieurs processus correspondent, ils sont tous tués (**TERM** par défaut).

Lorsque l'on veut envoyer un signal particulier, il faut le faire précéder d'un tiret.

Par ailleurs, si on veut que **pkill** cherche une chaîne de caractères sur la commande entière (y compris ses arguments), on peut fournir l'option **-f**. Par exemple, pour tuer « sauvagement » tout processus ayant la chaîne **toto.txt** dans la ligne de commandes, on utilise :

Terminal

```
$ pkill -9 -f toto.txt
```

Définition

Tous les processus fonctionnant sur un système UNIX ont un niveau de « gentillesse » : plus ce niveau est élevé, plus ils laissent le processeur disponible pour les autres processus. Ce niveau est le niceness et va de -20 à 20. Souvent, ce niveau est à 0 par défaut.

Certains processus ont besoin d'être plus réactifs que les autres, on peut alors réduire leur niceness grâce à la commande `renice`. Lorsqu'un processus peut passer derrière les autres, on peut augmenter son niceness. La commande `nice` permet de démarrer un processus avec un niceness donné.

VIRTUALBOX EN LIGNE DE COMMANDES

La virtualisation informatique permet de faire fonctionner plusieurs systèmes (logiciels) sur un seul système (matériel). Plusieurs solutions existent pour ce faire, VirtualBox étant l'une d'entre elles. Connu surtout pour son interface graphique facile à utiliser sur un poste de travail, VirtualBox peut aussi être utilisé en ligne de commandes...

En ligne de commandes, on manipule VirtualBox avec la commande **VBoxManage**. Celle-ci permet d'effectuer l'intégralité des actions que l'on peut retrouver dans l'interface graphique (création de machine virtuelle, assignation de disques virtuels, de CD, d'interfaces réseau, de périphériques USB, contrôle des machines en fonctionnement, etc.) auxquelles s'ajoutent les possibilités suivantes :

- Exécution en affichage déporté (pas de fenêtre localement, utilisation du protocole VNC ou RDP), ce mode est appelé *headless* ;
- Configuration des redirections de ports réseau de l'hôte vers la machine virtuelle ;
- Assignation d'un disque ou d'une partition physique à la machine virtuelle ;
- Migration à chaud vers un autre hôte (même lorsque le système d'exploitation hôte est différent)...

Nous aborderons ici des usages assez classiques de VirtualBox, à vous d'approfondir si vous voulez vous servir de ces fonctionnalités avancées ; la documentation officielle de VirtualBox (**https://www.virtualbox.org/manual/**) est très détaillée.

L'utilitaire **VBoxManage** accepte en premier argument une commande, suivie de différents arguments : **VBoxManage <commande> <arguments>**.

1. CRÉER UNE MACHINE VIRTUELLE

Commençons par créer une machine virtuelle ; pour cela, on utilise la commande **createvm**, qui prend au minimum l'argument **--name** :

Terminal
```
$ VBoxManage createvm --name "Ma super VM"
```

Une fois une machine virtuelle créée, il faut l'enregistrer, avec la commande **registervm** :

Terminal
```
$ VBoxManage registervm '/home/utilisateur/VirtualBox VMs/Ma
super VM/Ma super VM.vbox'
```

On peut également demander à VirtualBox d'enregistrer la machine virtuelle dès sa création, avec l'argument **--register**, on tape ainsi une seule commande au lieu de deux :

Terminal
```
$ VBoxManage createvm --name "Ma super VM" --register
```

Pour enlever l'enregistrement d'une machine virtuelle (pour qu'elle ne soit plus connue de VirtualBox mais que ses fichiers ne soient pas effacés), on peut utiliser la commande **unregistervm** :

Terminal
```
$ VBoxManage unregistervm "Ma super VM"
```

Lorsqu'elle est créée avec **createvm**, une machine virtuelle n'a que des caractéristiques de base : un processeur, un peu de mémoire, une carte graphique... Il faut configurer la machine pour qu'elle soit vraiment utilisable...

À savoir

VirtualBox a été créé il y a de nombreuses années par la société allemande Innotek GmbH ; la première version sous licence GPL est sortie en 2007. Innotek a ensuite été rachetée par Sun en 2008, elle-même rachetée par Oracle en 2010.

À savoir

On peut également choisir de créer une machine virtuelle via l'interface graphique pour ensuite la gérer en ligne de commandes : ces deux approches ne sont pas cloisonnées, les machines virtuelles et les fonctionnalités accessibles sont exactement les mêmes.

Bien que ça ne soit pas indispensable, on peut indiquer le système d'exploitation qui sera utilisé. Listons d'abord tous les systèmes d'exploitation connus, avec la commande `list ostypes` :

Terminal

```
$ VBoxManage list ostypes
```

Cette commande retourne plusieurs dizaines de systèmes, la valeur du champ `ID` sera à utiliser.

1.1 Interfaces réseau

Il peut y avoir jusqu'à 8 interfaces sur une machine virtuelle (seules les quatre premières peuvent être configurées dans l'interface graphique). Avec VirtualBox, on peut créer différents types de connexions réseau :

- ⇨ `nat` : la carte réseau virtuelle est automatiquement NATée, sur un réseau virtuel indépendant, avec un serveur DHCP virtuel fourni par VirtualBox (c'est le choix par défaut lorsque l'on crée une machine virtuelle par l'interface graphique) ;
- ⇨ `natnetwork` (disponible à partir de VirtualBox 4.3) : la carte réseau virtuelle est NATée, avec un réseau virtuel sur lequel plusieurs machines virtuelles peuvent être placées, VirtualBox fournissant un serveur DHCP ;
- ⇨ `bridged` : la carte réseau virtuelle communique directement avec un réseau physique, au travers de l'interface physique de l'hôte, elle est visible des autres machines du réseau ;
- ⇨ `intnet` : la carte réseau virtuelle est connectée à un réseau virtuel interne ; ce réseau interne peut accueillir plusieurs machines virtuelles mais pas l'hôte ;
- ⇨ `hostonly` : la carte réseau virtuelle est connectée à un réseau virtuel interne, auquel l'hôte a également accès grâce à une interface réseau virtuelle mise en place dynamiquement par VirtualBox.

1.2 Configuration de la machine virtuelle

Pour configurer les éléments de base d'une machine virtuelle, on utilise la commande `modifyvm`. Celle-ci prend différents arguments, selon les paramètres que l'on souhaite modifier. Le tableau ci-dessous en indique quelques-uns :

Argument	Paramètre
`--ostype <type>`	Type de système d'exploitation
`--memory <taille>`	Taille de la mémoire vive (en Mo)
`--vram <taille>`	Taille de la mémoire graphique (en Mo)
`--acpi on|off`	(Dés)activer l'ACPI
`--ioapic on|off`	(Dés)activer les IO-APIC (nécessaires pour Windows 64 bits)
`--accelerate3d on|off`	Accélération 3D
`--accelerate2dvideo on|off`	Accélération vidéo DirectDraw
`--nic<X> <type>`	Type d'interface réseau (`<X>` est un numéro de 1 à 8)

Cette commande accepte en réalité plus d'une centaine d'arguments !
Vous pouvez en obtenir la liste en tapant simplement, sans argument :

Terminal

```
$ VBoxManage modifyvm
```

Nous allons paramétrer notre machine virtuelle pour qu'elle accueille Windows 8 en 64 bits, avec 2 Go de RAM, 128 Mo de mémoire virtuelle, les accélérations 2D et 3D, ainsi qu'une interface réseau en NAT :

Terminal
```
$ VBoxManage modifyvm "Ma super VM" --ostype Windows8_64 --ioapic on
--memory 2048 --vram 128 --accelerate3d on --accelerate2dvideo on --nic1 nat
```

Cette commande ne retourne rien ; cependant, on peut accéder à ces informations détaillées avec la commande **showvminfo** :

Terminal
```
$ VBoxManage showvminfo "Ma super VM"
```

1.3 Disques virtuels

Pour créer un disque virtuel, on utilise la commande **createhd**. Celle-ci prend différents arguments, dont **--filename** pour le nom du fichier et **--size** pour la taille du disque virtuel :

Terminal
```
$ VBoxManage createhd --filename /home/utilisateur/VirtualBox\
VMs/Ma\ super\ VM/Ma\ super\ VM.vdi --size 10240
```

On peut également dupliquer un disque existant avec la commande **clonehd**, voire convertir une image brute (comme un disque physique ou une image extraite avec l'utilitaire **dd**), avec la commande **convertfromraw**.

Une fois le disque dur créé, il faut l'associer avec notre machine virtuelle, grâce à la commande **storageattach**. Il faut cependant d'abord créer un contrôleur de disque, avec la commande **storagectl** :

Terminal
```
$ VBoxManage storagectl "Ma super VM" --name "Controleur de disque" --add sata
$ VBoxManage storageattach "Ma super VM" --storagectl "Controleur de disque"
--port 0 --device 0 --type hdd --medium "/home/utilisateur/VirtualBox VMs/Ma
super VM/Ma super VM.vdi"
```

On va également attacher l'image ISO du disque d'installation de Windows 8, grâce à la même commande, mais avec des arguments différents :

Terminal
```
$ VBoxManage storageattach "Ma super VM" --storagectl "Controleur de disque"
--port 1 --device 0 --type dvddrive --medium "/home/utilisateur/Windows8.iso"
```

Avec **showvminfo**, on trouve alors la trace de ces associations.

2. MANIPULER LES MACHINES VIRTUELLES

Maintenant que la machine virtuelle est créée, on peut la démarrer, l'arrêter, s'y connecter… l'utiliser au quotidien, tout simplement.

2.1 Identifier les machines virtuelles

Commençons par obtenir la liste des machines virtuelles disponibles, avec la commande **list vms** :

Terminal
```
$ VBoxManage list vms
```

Les valeurs entre guillemets retournées par la commande sont les noms des machines virtuelles, celles entre accolades sont leurs identifiants uniques (UUID).

À savoir

VirtualBox accepte plusieurs formats de disques durs virtuels, parmi lesquels : *Virtual Disk Image* (VDI), le format spécifique à VirtualBox (par défaut) ; *Virtual Machine Disk* (VMDK), le format de VMware ; *Virtual Hard Disk* (VHD), le format de Microsoft. Il est toutefois préférable de rester sur le format de VirtualBox, pour des performances maximales ; on peut toujours convertir un disque virtuel a posteriori si nécessaire.

À retenir

Bien que plus simples à retenir, les noms des machines ont un inconvénient : ils peuvent changer. Les UUID, quant à eux, ne changeront jamais. Si vous souhaitez manipuler des machines virtuelles de manière automatique via des scripts, ou si vous souhaitez créer une nouvelle interface utilisateur au-dessus de ces commandes, vous avez tout intérêt à utiliser les UUID !

En réalité, d'autres éléments peuvent être listés, avec la commande **`VboxManage list <type>`**, où **`<type>`** est :

- ⇨ **`runningvms`** : les machines virtuelles en fonctionnement ;
- ⇨ **`ostypes`** : les types de système d'exploitation connus ;
- ⇨ **`hostdvds`**, **`hostfloppies`** : les disques amovibles (CD, DVD, disquettes) disponibles sur l'hôte ;
- ⇨ **`bridgedifs`**, **`hostonlyifs`**, **`dhcpservers`** : différents détails sur les réseaux virtuels ;
- ⇨ **`hostinfo`** : informations techniques sur l'hôte (processeurs, mémoire vive...) ;
- ⇨ **`hddbackends`** : types de disques virtuels supportés par VirtualBox ;
- ⇨ **`hdds`**, **`dvds`**, **`floppies`** : disques virtuels enregistrés auprès de VirtualBox ;
- ⇨ **`usbhost`** : les périphériques USB branchés sur l'hôte et non utilisés, qui peuvent être connectés aux machines virtuelles ;
- ⇨ **`usbfilters`** : filtres USB enregistrés dans VirtualBox (permettant l'attribution de périphériques USB physiques aux machines virtuelles) ;
- ⇨ **`systemproperties`** : propriétés globales de VirtualBox (notamment les limites pour chaque type de matériel virtuel) ;
- ⇨ **`extpacks`** : packs d'extensions installés.

Rappelons-nous également de la commande permettant d'obtenir des informations détaillées sur une machine virtuelle : **`showvminfo`**.

2.2 Démarrer une machine virtuelle

Pour lancer une machine virtuelle, la commande à utiliser est **`VboxHeadless`**, avec l'argument **`--startvm`** :

Terminal
```
$ VBoxHeadless --startvm "Ma super VM"
[...]
VRDE server is listening on port 3389.
```

On peut alors se connecter au port 3389 local, avec un client RDP (par exemple **`rdesktop`**).

Si on préfère utiliser le protocole VNC, il faut modifier le paramétrage de VirtualBox avec la commande suivante :

Terminal
```
$ VBoxManage setproperty vrdeextpack VNC
```

Il faut également préciser le mot de passe spécifique à cette machine virtuelle :

Terminal
```
$ VBoxManage modifyvm "Ma super VM" --vrdeproperty VNCPassword=<mot de passe>
```

Une fois cette commande exécutée, on peut démarrer **`VBoxHeadless`** :

Terminal
```
$ VBoxHeadless --startvm "Ma super VM"
[...]
VRDE server is listening on port 5900.
```

Cette fois-ci, on peut utiliser un client VNC pour se connecter à cette machine virtuelle.

On notera également que l'on peut fournir à **`VBoxHeadless`** l'option **`-c`** ou **`--capture`**, afin d'enregistrer l'affichage dans un fichier vidéo ; la documentation de VirtualBox explique en détails comment l'utiliser, vous pouvez aussi exécuter **`VBoxHeadless`** sans argument pour avoir certains détails.

En essayant cette commande, vous aurez remarqué qu'elle ne rend pas la main : tant que la machine virtuelle tourne, la console est inutilisable (sauf à mettre **VBoxHeadless** en arrière-plan avec la combinaison de touches [Ctrl]+[Z], puis la commande **bg** bien sûr, mais le processus reste « rattaché » à la console).

On peut utiliser l'option **--type headless** afin d'avoir à nouveau la main sur la console ou le terminal dès que la machine virtuelle est démarrée :

Terminal
```
$ VBoxManage startvm "Ma super VM" --type headless
```

En exécutant cette commande, on remarque que l'affichage de la machine virtuelle est totalement inaccessible (ni RDP, ni VNC en écoute) : en effet, on n'a pas configuré ses paramètres d'accès distant. On utilise alors la commande **modifyvm** avec l'option **--vrde** pour activer l'accès distant. Différentes options existent, en complément, pour configurer l'accès distant ; on utilisera ici **--vrdeport** pour configurer le port d'écoute, ainsi que **--vrdeextpack** pour forcer l'écoute en VNC, même si on change ultérieurement la méthode par défaut :

Terminal
```
$ VBoxManage modifyvm "Ma super VM" --vrde on --vrdeport 5900 --vrdeextpack VNC
```

On peut alors démarrer la commande avec **startvm** et l'option **--type headless** : cette fois-ci, VirtualBox sera en écoute sur le port 5900 pour cette machine virtuelle.

Bien sûr, si on veut une machine sans écran, à laquelle on n'accéderait qu'en SSH par exemple, on peut désactiver l'accès distant avec **--vrde off**.

2.3 Arrêter une machine virtuelle

Pour arrêter une machine virtuelle, l'idéal est d'arrêter proprement le système d'exploitation qu'elle contient : pour peu qu'il supporte l'ACPI (et que ça ne soit pas désactivé dans VirtualBox), il saura indiquer qu'il est éteint et la machine virtuelle s'arrêtera d'elle-même. Cependant, on peut manipuler l'alimentation de la machine avec quelques options de la commande **controlvm** :

- ⇨ **pause** : arrêter temporairement la machine virtuelle (sans la fermer) ;
- ⇨ **resume** : reprendre après l'utilisation de la commande **pause** ;
- ⇨ **reset** : redémarrer brutalement la machine virtuelle ;
- ⇨ **poweroff** : arrêter brutalement la machine virtuelle ;
- ⇨ **acpipowerbutton** : envoyer au système d'exploitation l'information comme quoi le bouton d'alimentation a été appuyé ;
- ⇨ **savestate** : enregistrer l'état de la machine virtuelle sur disque dur et l'arrêter (lors du prochain allumage, ces données seront récupérées et l'état sera restauré, au lieu de redémarrer le système entièrement) – cela ressemble à l'hibernation, qui est gérée par le système invité ;
- ⇨ **discardstate** : supprimer l'état enregistré (équivaut à un arrêt brutal de la machine virtuelle).

Par exemple, pour arrêter « brutalement » notre machine virtuelle, la commande est :

Terminal
```
$ VBoxManage controlvm "Ma super VM" poweroff
```

2.4 Effectuer des instantanés

VirtualBox permet de capturer des instantanés des machines virtuelles : un état précis, qui peut être rétabli ultérieurement en annulant toutes les modifications postérieures, par exemple pour revenir en arrière lors de l'installation d'un logiciel. Les manipulations sur les instantanés se font avec la commande **snapshot**, dont les possibilités sont :

- ⇨ **VBoxManage snapshot <mv> take <nom>** : prendre un instantané de la machine **<mv>** en le nommant **<nom>** ;

- ⇨ **VBoxManage snapshot <mv> delete <nom>** : supprimer l'instantané **<nom>** (tout en conservant l'état actuel de la machine virtuelle) ;
- ⇨ **VBoxManage snapshot <mv> restore <nom>** : restaurer l'instantané **<nom>** (l'état actuel de la machine est alors perdu) ;
- ⇨ **VBoxManage snapshot <mv> restorecurrent** : restaurer le dernier instantané (l'état actuel est perdu).

Par exemple, imaginons que Windows 8 est installé et que l'on veuille y installer LibreOffice, mais pouvoir revenir en arrière si cela ne convient pas. Pendant que la machine est en fonctionnement, on crée un instantané :

Terminal
```
$ VBoxManage snapshot "Ma super VM" take "Installation LibreOffice"
```

On peut vérifier la présence de cet instantané avec **showvminfo** :

Terminal
```
$ VBoxManage showvminfo "Ma super VM"
[...]
Snapshots:
   Name: Installation LibreOffice (UUID: aa374394-955e-49db-8a13-73d204078140) *
```

On installe ensuite LibreOffice, on vérifie que cela fonctionne, on l'évalue. Si l'installation a échoué (ou si LibreOffice ne convient pas à l'utilisateur final), il suffit d'arrêter la machine virtuelle puis de restaurer le dernier instantané :

Terminal
```
$ VBoxManage controlvm "Ma super VM" poweroff
$ VBoxManage snapshot "Ma super VM" restorecurrent
```

Qu'on soit retourné sur l'instantané (problème d'installation), ou qu'on ait choisi de conserver le système tel qu'il a évolué (LibreOffice installé), on peut supprimer cet instantané (même si la machine est en train de fonctionner) :

Terminal
```
$ VBoxManage snapshot "Ma super VM" delete "Installation LibreOffice"
```

2.5 Partager des dossiers de l'hôte

VirtualBox permet de partager des dossiers du système d'exploitation hôte avec les machines virtuelles. En ligne de commandes, les partages se gèrent avec la commande **sharedfolder** et se créent lorsque la machine virtuelle est arrêtée. Pour ajouter un partage :

Terminal
```
$ VBoxManage sharedfolder add "Ma super VM" --name Temp --hostpath /tmp
```

Les options suivantes peuvent être ajoutées :

- ⇨ **--transient** : le partage n'est que temporaire, il disparaîtra au prochain redémarrage (cette commande doit alors être exécutée pendant que la machine virtuelle fonctionne) ;
- ⇨ **--readonly** : la machine virtuelle n'a pas le droit d'écrire dans les fichiers partagés, ni de les effacer ;
- ⇨ **--automount** : monte automatiquement le partage sur le système virtualisé (les « additions invité » doivent être installées).

Pour supprimer un partage, la commande est :

Terminal
```
$ VBoxManage sharedfolder remove "Ma super VM" --name Temp
```

Cette commande doit être exécutée machine virtuelle arrêtée. Et comme pour la création, on peut utiliser l'option **--transient** pour traiter un partage temporaire, auquel cas la machine virtuelle doit être en fonctionnement.

Pour utiliser ce partage, s'il n'a pas été monté automatiquement, les méthodes peuvent être les suivantes :

⇨ sous Windows :

Terminal
```
> NET USE X: \\vboxsvr\<nom du partage>
```

⇨ sous Linux :

Terminal
```
$ sudo mount -t vboxsf <nom du partage> <point de montage>
```

⇨ sous Solaris :

Terminal
```
# mount -t vboxfs <nom du partage> <point de montage>
```

2.6 Importer/exporter une machine virtuelle

VirtualBox est capable d'ouvrir une machine enregistrée au format OVF (*Open Virtualization Format*), ainsi que de créer de telles machines. Ce format permet d'échanger des applications virtuelles (*virtual appliances*) sans se soucier du logiciel de virtualisation que le destinataire utilise. Les commandes **import** et **export** permettent d'importer et d'exporter des machines virtuelles dans ce format.

Par exemple, pour exporter notre machine virtuelle, on utilisera :

Terminal
```
$ VBoxManage export "Ma super VM" --output MaSuperVM.ovf
```

On peut alors la distribuer sous cette forme, avec les fichiers ***.ovf** et ***.vmdk** liés. Il est également possible de placer la configuration de la machine virtuelle (**ovf**) et le(s) disque(s) virtuel(s) (**vmdk**) dans un même fichier, grâce au format OVA :

Terminal
```
$ VBoxManage export "Ma super VM" --output MaSuperVM.ova
```

Pour importer une machine virtuelle, la commande est encore plus simple :

Terminal
```
$ VBoxManage import UneAutreVM.ova
```

3. POUR ALLER PLUS LOIN…

Nous n'avons vu ici qu'une partie des possibilités de VirtualBox en ligne de commandes : en réalité, on s'est concentré sur l'utilisation de base, sans aucune fonctionnalité exotique. Pour approfondir le sujet, vous pouvez consulter le chapitre 8 de la documentation officielle, titré très justement *VBoxManage* (**https://www.virtualbox.org/manual/ch08.html**). Vous pourrez alors paramétrer vos machines virtuelles pour utiliser des périphériques USB, des cartes PCI, des disques durs physiques, faire des transferts de machines en fonctionnement d'un hôte à un autre… Toutes ces choses que certains logiciels propriétaires au coût exorbitant se targuent d'être les seuls à bien faire.

Vous pouvez aussi vous intéresser à **libvirt**, qui offre des « bindings » dans différents langages de programmation afin de manipuler de nombreux systèmes de virtualisation, dont VirtualBox (**http://libvirt.org**). Enfin, vous pouvez vous tourner vers phpVirtualBox pour bénéficier d'une interface web permettant de gérer des parcs de machines virtuelles avec VirtualBox (**http://sourceforge.net/projects/phpvirtualbox**). ▮

UN STOCKAGE PLUS « SOUPLE » AVEC LOGICAL VOLUME MANAGER

LVM (Logical Volume Manager) est un outil permettant de gérer des espaces de stockage de manière très flexible. Prenons deux exemples. Le premier, vous avez un disque de 4 To mais ne savez pas comment le partitionner : avec LVM, vous pouvez créer des petits volumes, que vous agrandirez au fur et à mesure, selon vos besoins. Le second, vous avez 3 ou 4 disques de 120 Go et voulez vous affranchir d'une gestion compliquée des points de montage : vous les fusionnez dans un seul espace LVM et pouvez créer un ou plusieurs volume(s) qui s'étend(ent) automatiquement sur plusieurs d'entre eux.

Nous allons commencer par éclaircir les concepts liés aux disques durs de manière générale, pour ensuite introduire LVM. Quand tout sera bien clair, nous allons créer, supprimer, agrandir des volumes LVM. Enfin, nous découvrirons qu'il est possible d'installer un système entier avec LVM, pour profiter pleinement de sa flexibilité.

Dans les exemples qui vont suivre, nous travaillons sur une machine virtuelle, sur laquelle on a installé Ubuntu Linux en version 12.04. Cette machine virtuelle est dotée de trois disques durs (virtuels) de 8 Go chacun, le système étant installé sur une partition de 5 Go créée sur le premier de ces trois disques. Bien sûr, votre cas sera très différent, à vous d'adapter les commandes comme vous le souhaitez !

La méthode fonctionnera sur n'importe quelle distribution Linux, LVM étant un outil universel et non limité à Ubuntu.

1. NOTIONS DE BASE

Avant de parler de LVM, comprenons comment s'organise un disque dur. Avec l'utilitaire **fdisk** ou **parted**, on peut voir les partitions créées sur les disques du système ; ce sujet a déjà été évoqué dans un précédent article de ce même numéro.

En l'occurrence, la commande **parted** donne le résultat suivant :

Terminal

```
$ sudo parted -l
Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sda : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début   Fin     Taille  Type     Système de fichiers  Fanions
 1      1049kB  5120MB  5119MB  primary  ext4                 démarrage

Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sdb : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début  Fin  Taille  Type  Système de fichiers  Fanions

Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sdc : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début  Fin  Taille  Type  Système de fichiers  Fanions
```

On a donc bien trois disques durs, dont seul le premier a une partition, qui fait environ 5 Go.

Sur un PC pas très récent, avec un BIOS, les systèmes de partitionnement MSDOS ne supportent qu'un maximum de 4 partitions primaires. L'une d'entre elles peut être une partition étendue, contenant des sous-partitions logiques, dont le nombre est illimité (cependant, les outils logiciels peuvent limiter ce nombre à quelques dizaines). Les PC vendus depuis quelques temps ont un UEFI au lieu d'un BIOS, ils supportent alors le partitionnement GPT qui permet 128 partitions principales.

Une partition, ce n'est rien d'autre qu'un découpage du disque, noté dans une table quelque part sur ce même disque. Il faut ensuite définir un format de stockage pour les données que

l'on place dans une partition, il s'agit des systèmes de fichiers. Lorsque l'on dit que l'on *formate* une partition, on y place un système de fichiers vierge. La perte de données n'est qu'un effet de bord de ce formatage, les données elles-mêmes n'étant pas effacées, elles sont juste déréférencées ; c'est pour cela que certains outils, comme **photorec**, sont capables de retrouver des données sur une partition formatée accidentellement.

Il existe plusieurs types de systèmes de fichiers pour les partitions. Sous Linux, les plus connus sont Ext2, son successeur Ext3 et le successeur de celui-ci, utilisé dans les distributions actuelles, Ext4. D'ici quelques années, Btrfs (prononcer « ButterFS ») aura probablement remplacé Ext4, mais il est à l'heure actuelle encore en développement (bien que déjà particulièrement stable) : il vaut mieux éviter de l'utiliser sur une machine en production. Sous Microsoft Windows, on retrouve NTFS et, de plus en plus rarement, FAT. Dans le monde d'Apple, ce sont les systèmes de fichiers HFS et HFS+ qui sont utilisés.

Commençons par créer une nouvelle partition sur le premier disque dur ; on va utiliser **fdisk** pour cela (on découvrira la méthode avec **parted** plus tard) :

Terminal

```
$ sudo fdisk /dev/sda

Commande (m pour l'aide) : n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Numéro de partition (1-4, 2 par défaut) : 2
Premier secteur (10000384-16777215, 10000384 par défaut) : 10000384
Dernier secteur, +secteurs ou +taille{K,M,G} (10000384-16777215, 16777215
par défaut) : 16777215

Commande (m pour l'aide) : w
La table de partitions a été altérée.

Appel d'ioctl() pour relire la table de partitions.

Attention : la table de partitions n'a pas pu être relue : erreur 16 :
Périphérique ou ressource occupé.
Le noyau continue à utiliser l'ancienne table. La nouvelle sera utilisée
lors du prochain démarrage ou après avoir exécuté partprobe(8) ou
kpartx(8).
Synchronisation des disques.
$ sudo partprobe
```

Créons ensuite un système de fichiers ext4 sur cette partition :

Terminal

```
$ sudo mkfs.ext4 /dev/sda2
[...]
Écriture des superblocs et de l'information de comptabilité du
système de
fichiers : complété
```

Dans le cadre d'une utilisation classique d'un disque dur, celui-ci est découpé en partitions fixes une fois pour toutes : les partitions peuvent difficilement évoluer. Vous écrivez ensuite directement dans la partition, en vous conformant à l'organisation des fichiers du système de fichiers choisi. C'est simple, mais rigide.

Définition

Le BIOS (*Basic Input-Output System*) est un ensemble de services permettant de faire partiellement abstraction de la couche matérielle, afin d'offrir aux systèmes d'exploitation des méthodes communes sur des matériels différents. Il existe depuis les années 70. C'est le premier élément en œuvre dans le démarrage d'un ordinateur.

Depuis 2010, le BIOS est progressivement remplacé par l'UEFI (*Unified Extensible Firmware Interface*), qui lève des limitations liées au BIOS, notamment le nombre de partitions primaires ou l'impossibilité de démarrer sur un disque de plus de 2 To. De plus, l'UEFI est indépendant de l'interface matérielle (processeur), contrairement au BIOS qui reste lié aux plateformes x86 : la gestion du démarrage sur des plateformes différentes est donc facilitée.

2. LA SOUPLESSE DE LVM

LVM est un gestionnaire de volumes logiques. Il s'intercale entre les disques et les systèmes de fichiers et permet de virtualiser la gestion des disques.

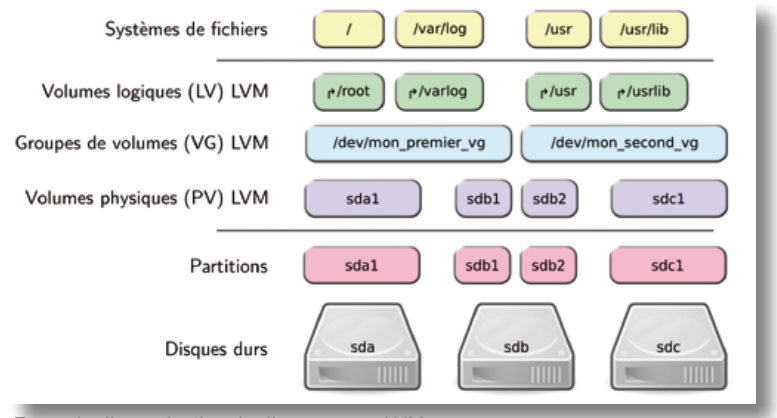
On le représente en trois couches ; commençons par celle du milieu : elle s'appelle **VG**, pour *Volume Group*. Un groupe de volumes, c'est comme un disque virtuel dynamique. Au-dessus de cette couche se trouvent les **LV**, pour *Logical Volumes*. Un volume logique, c'est comme une partition dynamique. On peut créer autant de volumes logiques que l'on veut dans un groupe de volumes. On peut les redimensionner à chaud (les agrandir ou les diminuer) et bien évidemment, les supprimer très facilement. La couche du bas, c'est celle qui permet de faire coller le virtuel du VG à la réalité du ou des disques physiques. Un VG s'appuie sur un ou plusieurs **PV**, pour *Physical Volumes*. Un volume physique, c'est simplement un format spécial que l'on donne à une partition physique du disque. On associe alors un ou plusieurs PV à un VG, après quoi on s'affranchit totalement de la couche physique, on ne travaille plus qu'au niveau virtuel du VG.

Pour résumer :

- Sur un disque sans LVM, on a des partitions, formatées avec des systèmes de fichiers, dans lesquels on met des fichiers ;
- Sur un disque avec LVM, on a des partitions, sur lesquelles sont placés des PV, regroupés au sein d'un ou plusieurs VG, dans lesquels sont créés des LV, formatés avec des systèmes de fichiers, dans lesquels on met des fichiers.

Sur un ordinateur géré par LVM, on retrouve parfois une partition pour `/boot` ou la racine en dehors de LVM, et tout le reste du disque dédié à LVM. Cela est lié au fait que les anciennes versions de GRUB, le chargeur de démarrage, n'était pas capable de démarrer sur un volume LVM ; GRUB a évolué, mais chez certains, l'habitude est restée. On peut maintenant avoir une seule partition sur le disque, contenant un PV. Ce PV est inclus dans un VG, dans lequel peuvent être créés différents LV : un pour la racine, un pour `/home`, un pour le swap, etc.

Une taille optimale (ou minimale) est donnée à chacun de ces volumes et le reste du VG reste libre pour pouvoir agrandir les LV ultérieurement, quand le besoin s'en fait sentir et de façon adaptée à celui-ci. Sur un ordinateur géré avec LVM, tout l'espace disque n'est donc pas alloué initialement aux partitions et vous n'avez pas besoin de vous demander dès le début quel partitionnement sera mieux adapté à ce que vous ferez dans 6 mois, un an ou plus...



► *Exemple d'organisation de disques avec LVM*

À retenir

De nombreuses distributions référencent les partitions par leur UUID (identifiant numérique unique) dans le fichier `/etc/fstab`, qui contient la liste des partitions à monter au démarrage. Lors du formatage, l'UUID d'une partition change. Par conséquent, si vous formatez une partition déjà référencée, il faut penser à modifier son UUID dans `/etc/fstab`. Pour obtenir la liste des UUID des partitions du système, il faut utiliser la commande `sudo blkid`.

3. DÉCOUVRONS LVM

Installons tout d'abord les outils permettant de gérer des volumes LVM. Sous Debian et Ubuntu, ils sont contenus dans le paquet **lvm2** :

Terminal
```
$ sudo apt-get install lvm2
```

Nous partirons du principe que les disques du système ne sont pas encore gérés par LVM : dans notre exemple, nous allons créer des volumes LVM à partir d'espaces vierges. Si vous voulez expérimenter sur votre ordinateur, n'oubliez pas que ces manipulations nécessitent de formater les partitions, donc d'effacer les données existantes !

3.1 Possibilité 1 : simuler des disques

Si vous n'avez pas d'espace disponible sur votre ou vos disque(s) dur(s), vous pouvez utiliser des fichiers afin de simuler des partitions de disques, comme ce que l'on fait avec les images ISO pour les images de CD, de DVD, etc.

Commençons d'abord par créer des fichiers vierges, avec la commande **dd** :

Terminal
```
$ dd if=/dev/zero of=image_disque1.img bs=1M count=500
500+0 enregistrements lus
500+0 enregistrements écrits
524288000 octets (524 MB) copiés, 1,12757 s, 465 MB/s
$ dd if=/dev/zero of=image_disque2.img bs=1M count=500
500+0 enregistrements lus
500+0 enregistrements écrits
524288000 octets (524 MB) copiés, 3,75317 s, 140 MB/s
$ dd if=/dev/zero of=image_disque3.img bs=1M count=500
500+0 enregistrements lus
500+0 enregistrements écrits
524288000 octets (524 MB) copiés, 4,09711 s, 128 MB/s
```

Ici, on a créé trois fichiers de 500 Mo (500 blocs de 1 Mo), à partir du générateur de zéros (**/dev/zero**). Attention à ne pas intervertir les arguments **if** et **of** quand vous utilisez cette commande, cela pourrait être désastreux !

On pourrait créer plusieurs partitions dans une seule de ces images, mais leur manipulation serait très compliquée. On va alors utiliser directement chacun de ces trois fichiers comme si c'était trois partitions. Pour cela, on utilise la commande **losetup**, afin qu'ils soient vus comme trois volumes physiques, avec des périphériques associés dans **/dev** :

Terminal
```
$ sudo losetup /dev/loop0 image_disque1.img
$ sudo losetup /dev/loop1 image_disque2.img
$ sudo losetup /dev/loop1 image_disque3.img
```

Vous pourrez alors remplacer les occurrences de **/dev/sda2**, **/dev/sdb1** et **/dev/sdc1** par **/dev/loop0**, **/dev/loop1** et **/dev/loop2** dans les commandes qui suivront.

Attention, une fois que vous aurez fini d'utiliser ces vrais-faux volumes, n'oubliez pas de les dissocier avec les commandes suivantes :

Terminal
```
$ sudo losetup -d /dev/loop0
$ sudo losetup -d /dev/loop1
$ sudo losetup -d /dev/loop2
```

3.2 Possibilité 2 : préparer des disques libres

Nous avons vu comment créer des images de disques dans des fichiers, mais dans notre exemple, nous travaillerons directement sur de « vraies » partitions, grâce aux trois disques associés à notre machine virtuelle. Bien sûr, ceux qui ont créé des images dans des fichiers peuvent sauter ce chapitre.

Nous avons déjà créé une partition **/dev/sda2** plus haut, nous allons maintenant créer une partition dans chacun des disques **/dev/sdb** et **/dev/sdc**. Pour cela, nous allons utiliser – comme promis – **parted**. Nous utilisons avec **/dev/sdb** une méthode interactive et avec **/dev/sdc** une méthode en une commande :

Terminal

```
$ sudo parted /dev/sdb
GNU Parted 2.3
Utilisation de /dev/sdb
Bievenue sur GNU Parted ! Tapez 'help' pour voir la liste des commandes.
(parted) print
Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sdb : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début  Fin  Taille  Type  Système de fichiers  Fanions

(parted) mkpart
Type de partition ?  primary/primaire/extended/étendue? primary
Type de système de fichiers ?  [ext2]? ext2
Début ? 0
Fin ? 8590
Avertissement: L'alignement de la partition ainsi définie n'est pas optimal au niveau performance.
Ignorer/Ignore/Annuler/Cancel? ignore
(parted) quit
Information: Ne pas oublier de mettre à jour /etc/fstab si nécessaire.

$ sudo parted --align optimal /dev/sdc mkpart primary ext2 0% 100%
Information: Ne pas oublier de mettre à jour /etc/fstab si nécessaire.
```

Enfin, on vérifie la structure des disques :

Terminal

```
$ sudo parted -l
Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sda : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début   Fin     Taille  Type     Système de fichiers  Fanions
 1      1049kB  5120MB  5119MB  primary  ext4                 démarrage
 2      5120MB  8590MB  3470MB  primary


Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sdb : 8590MB
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début  Fin     Taille  Type     Système de fichiers  Fanions
 1      512B   8590MB  8590MB  primary


Modèle: ATA VBOX HARDDISK (scsi)
Disque /dev/sdc : 8590MB
```

Terminal
```
Taille des secteurs (logiques/physiques): 512B/512B
Table de partitions : msdos

Numéro  Début   Fin     Taille  Type     Système de fichiers  Fanions
 1      1049kB  8590MB  8589MB  primary
```

On a bien les partitions **/dev/sda2**, **/dev/sdb1** et **/dev/sdc1**.

3.3 Physical Volumes

La première étape dans la mise en œuvre de LVM est le formatage des partitions comme volumes physiques. Cela se fait avec la commande **pvcreate** :

Terminal
```
$ sudo pvcreate /dev/sda2
Physical volume "/dev/sda2" successfully created
$ sudo pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created
$ sudo pvcreate /dev/sdc1
Physical volume "/dev/sdc1" successfully created
```

Comme on le voit, la création de volumes physiques est plutôt simple. On peut vérifier leur existence avec la commande **pvs** :

Terminal
```
$ sudo pvs
PV         VG   Fmt  Attr PSize PFree
/dev/sda2       lvm2 a-   3,23g 3,23g
/dev/sdb1       lvm2 a-   8,00g 8,00g
/dev/sdc1       lvm2 a-   8,00g 8,00g
```

Notons l'existence des commandes suivantes :

- **pvchange** pour changer quelques propriétés du disque, notamment si on autorise LVM à y placer des données (on peut désactiver cela lorsque l'on veut se préparer à « sortir » un volume physique d'un groupe de volumes, par exemple – elle n'est cependant pas très utilisée) ;
- **pvck** pour vérifier la validité des métadonnées LVM sur un volume physique (on l'utilise très peu) ;
- **pvdisplay** pour afficher des informations détaillées sur les volumes physiques (données plus complètes qu'avec **pvs**) ;
- **pvmove** pour déplacer des données d'un volume physique vers un autre, au sein du même groupe de volumes (très peu utilisée également) ;
- **pvremove** pour supprimer les métadonnées du volume physique (il ne sera alors plus vu comme tel) ;
- **pvresize** pour redimensionner un volume physique, lorsque l'on agrandit une partition d'un disque par exemple ;
- **pvscan** pour rechercher tous les volumes physiques présents sur la machine.

3.4 Créer un VG

La création d'un groupe de volumes est également très simple ; la commande **vgcreate** attend comme seuls arguments le nom du groupe et les volumes physiques à y associer :

Terminal
```
$ sudo vgcreate mon_vg /dev/sda2 /dev/sdb1
Volume group "mon_vg" successfully created
```

On peut alors vérifier que ce groupe de volumes existe, soit avec la commande **vgs**, soit avec la commande **vgdisplay**. Avec l'option **-v**, cette dernière indique également les PV et les LV associés au VG :

Terminal

```
$ sudo vgs
VG     #PV #LV #SN Attr   VSize  VFree
mon_vg   2   0   0 wz--n- 11,23g 11,23g
$ sudo vgdisplay -v
  Finding all volume groups
  Finding volume group "mon_vg"
--- Volume group ---
  VG Name               mon_vg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               11,23 GiB
  PE Size               4,00 MiB
  Total PE              2874
  Alloc PE / Size       0 / 0
  Free  PE / Size       2874 / 11,23 GiB
  VG UUID               B3GswF-B1Pm-dS8X-vUkn-aOXS-19R4-QIlpJl

--- Physical volumes ---
  PV Name               /dev/sda2
  PV UUID               o4DWKj-3L0J-oc2R-r9dg-TAa6-Wyml-AAWTJl
  PV Status             allocatable
  Total PE / Free PE    827 / 827

  PV Name               /dev/sdb1
  PV UUID               o0dPlR-aV06-03pf-gp3W-bqwQ-lvvM-YkXoCs
  PV Status             allocatable
  Total PE / Free PE    2047 / 2047
```

On constate ici que le groupe de volumes s'appelle bien **mon_vg** (ouf !), qu'il a une taille totale (**VG Size**) de 11,23 Go, dont l'intégralité est disponible (**Free PE / Size**).

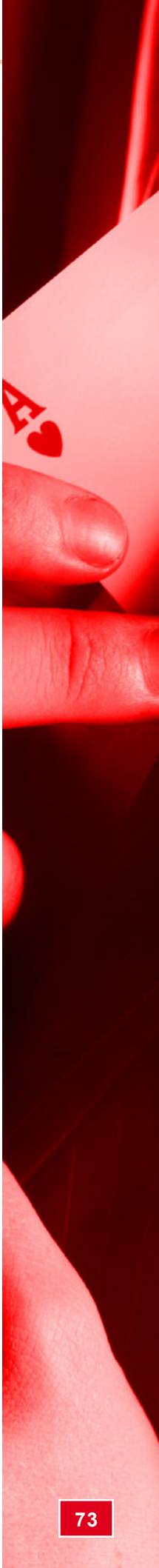
Notons une donnée : la taille du **PE**, pour *Physical Extent*. C'est la plus petite unité gérée par LVM : toute taille donnée dans le cadre de LVM est un multiple de cette taille-là ; ici, un PE fait 4 Mo, un volume logique pourra alors faire 32 Mo ou 36 Mo, mais il ne fera jamais 35,5 Mo…

À partir de là, on n'a plus besoin de parler des volumes physiques : exit les partitions, les disques, les images… Tout ce qui entre en jeu maintenant, c'est ce groupe de volumes et les volumes logiques que l'on va y créer.

3.5 Créer un LV

Nous voici avec notre VG, sorte de disque virtuel. Allons à la couche au-dessus, créons une partition virtuelle : un volume logique. Pour cela, on utilise la commande **lvcreate**, qui demande différentes informations. Dans son utilisation la plus classique, on lui donne :

- ⇨ la taille du LV à créer (avec l'option **-L**) ;
- ⇨ le nom du LV (avec l'option **-n**) ;
- ⇨ le VG à utiliser (comme dernier argument, sans préfixe).



Créons par exemple un volume logique appelé **super_lv**, avec une taille de 2 Go :

Terminal

```
$ sudo lvcreate -L 3G -n super_lv mon_vg
Logical volume "super_lv" created
```

De même qu'avec les PV et les VG, on peut lister les LV avec les commandes **lvs** et **lvdisplay** :

Terminal

```
$ sudo lvs
LV       VG     Attr   LSize Origin Snap%  Move Log Copy%  Convert
super_lv mon_vg -wi-a- 3,00g
$ sudo lvdisplay
--- Logical volume ---
  LV Name                /dev/mon_vg/super_lv
  VG Name                mon_vg
  LV UUID                S3JSlt-NpUH-KEUx-aRuZ-b99g-6ic2-pMRH1A
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                3,00 GiB
  Current LE             768
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

Dans la ligne **LV Name**, on retrouve le chemin par lequel ce volume logique est disponible. En effet, les volumes logiques LVM sont accessibles en tant que périphériques sous la forme **/dev/<VG>/<LV>**. On les retrouve également avec l'ensemble des autres éléments gérés par le *device mapper* du noyau Linux, dans **/dev/mapper**, en l'occurrence sous le nom **/dev/mapper/<VG>-<LV>**.

Utilisons la même commande pour créer un second volume logique :

Terminal

```
$ sudo lvcreate -L 147M -n autre_lv mon_vg
Rounding up size to full physical extent 148,00 MiB
Logical volume "autre_lv" created
```

Ici, la taille demandée n'était pas un multiple de la taille d'un PE : un arrondi a donc été automatiquement fait par **lvcreate**.

Maintenant que l'on a créé des LV, on peut s'interroger sur l'espace disponible restant : on l'obtient alors avec **vgs** ou **vgdisplay** :

Terminal

```
$ sudo vgs
VG     #PV #LV #SN Attr   VSize  VFree
mon_vg   2   2   0 wz--n- 11,23g 8,08g
```

Dans notre exemple, il reste un peu plus de 8 Go disponibles dans le LV.

Notons que, comme annoncé, nous n'avons pas une fois évoqué les disques : par défaut, on ne sait pas où sont situées ces données sur les disques. C'est pourquoi, il est important de mettre dans un même groupe de volumes des partitions placées sur des disques aux performances équivalentes.

3.6 Formater un LV

On explique depuis le début qu'un LV peut être vu comme une simple partition… Dans ce cas, pour le formater, ne faut-il pas simplement utiliser une commande que l'on connaît bien, **mkfs** ?

Terminal

```
$ sudo mkfs.ext4 /dev/mon_vg/super_lv
[...]
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété
```

Eh bien oui ! Ça fonctionne simplement de cette manière. On peut alors monter ce volume comme n'importe quelle partition et y écrire des données :

Terminal

```
$ sudo mount /dev/mon_vg/super_lv /mnt
$ sudo mkdir /mnt/petit_repertoire
$ sudo chown diamond.diamond /mnt/petit_repertoire
$ echo "Ça fonctionne..." > /mnt/petit_repertoire/hello.world
$ cat /mnt/petit_repertoire/hello.world
Ça fonctionne...
```

3.7 Agrandissement d'un LV

Nous voici au cœur du sujet, l'une des principales raisons d'utiliser LVM : la possibilité d'agrandir un volume. En réalité, une telle manipulation – complexe avec un partitionnement classique – est courante et très simple avec LVM ; on utilise alors la commande **lvextend** :

Terminal

```
$ sudo lvextend -L +1G /dev/mon_vg/super_lv
Extending logical volume super_lv to 4,00 GiB
Logical volume super_lv successfully resized
$ sudo lvextend -L 5G /dev/mon_vg/autre_lv
Extending logical volume autre_lv to 5,00 GiB
Logical volume autre_lv successfully resized
```

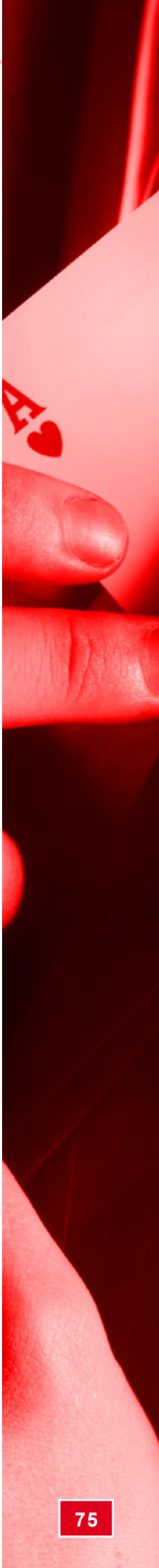
Avec l'option **-L**, on indique à **lvextend** la taille que l'on veut obtenir en octets : en la préfixant avec un **+**, on indique la quantité d'espace **à ajouter** ; sans préfixe, on indique **la taille cible**. L'argument suivant est évidemment le chemin vers le LV à agrandir.

Nos volumes logiques sont agrandis, pourtant l'espace disque disponible n'a pas augmenté :

Terminal

```
$ sudo lvs
LV       VG     Attr   LSize Origin Snap%  Move Log Copy%  Convert
autre_lv mon_vg -wi-a- 5,00g
super_lv mon_vg -wi-ao 4,00g
$ df -h /mnt
Sys. de fichiers            Taille Utilisé Dispo Uti% Monté sur
/dev/mapper/mon_vg-super_lv   2,9G    4,6M  2,8G   1% /mnt
```

En effet, le volume a été augmenté, mais on n'a pas touché au système de fichiers qui y est stocké : il faut également l'agrandir, avec la commande **resize2fs** pour les systèmes de fichiers Ext2, Ext3 et Ext4 :



Terminal

```
$ sudo resize2fs /dev/mon_vg/super_lv
resize2fs 1.42 (29-Nov-2011)
Le système de fichiers de /dev/mon_vg/super_lv est monté sur /mnt ; le changement de
taille doit être effectué en ligne
old_desc_blocks = 1, new_desc_blocks = 1
Le système de fichiers /dev/mon_vg/super_lv a maintenant une taille de 1048576 blocs.

$ df -h /mnt
Sys. de fichiers            Taille Utilisé Dispo Uti% Monté sur
/dev/mapper/mon_vg-super_lv   3,9G    6,0M  3,7G   1% /mnt
```

Et voilà ! L'espace disponible a été agrandi en deux commandes plutôt simples.

3.8 Diminuer la tailler d'un LV

Avec la commande **lvreduce**, on peut réduire la taille d'un volume logique. Par contre, il faut être très prudent, car il ne s'agirait pas de perdre des données présentes dans le système de fichiers ! Il faut donc d'abord réduire le système de fichiers, pour ensuite réduire le LV ; par ailleurs, pour être sûr de ne pas faire de bêtises, assurons-nous de laisser un peu de marge entre la taille du système de fichiers et la taille du volume logique.

Par contre, avec les systèmes de fichiers Ext2, Ext3 et Ext4, il n'est pas possible de réduire la taille à chaud : il faudra donc arrêter tout logiciel qui utilise cet espace disque, puis le démonter, le temps de le réduire... De plus, **resize2fs** nécessite que l'espace disque ait été vérifié par **e2fsck** auparavant. Cela nécessite beaucoup de temps, c'est bien plus contraignant...

Terminal

```
$ sudo umount /mnt
$ sudo e2fsck -f /dev/mon_vg/super_lv
e2fsck 1.42 (29-Nov-2011)
Passe 1 : vérification des i-noeuds, des blocs et des tailles
Passe 2 : vérification de la structure des répertoires
Passe 3 : vérification de la connectivité des répertoires
Passe 4 : vérification des compteurs de référence
Passe 5 : vérification de l'information du sommaire de groupe
/dev/mon_vg/super_lv : 13/262144 fichiers (0.0% non contigüs), 34384/1048576 blocs
$ sudo resize2fs /dev/mon_vg/super_lv 1900M
resize2fs 1.42 (29-Nov-2011)
En train de redimensionner le système de fichiers sur /dev/mon_vg/super_lv à
486400 (4k) blocs.
Le système de fichiers /dev/mon_vg/super_lv a maintenant une taille de 486400
blocs.
$ sudo mount /dev/mon_vg/super_lv /mnt
$ df -h /mnt
Sys. de fichiers            Taille Utilisé Dispo Uti% Monté sur
/dev/mapper/mon_vg-super_lv   1,8G    4,6M  1,7G   1% /mnt
```

On constate ici que notre système de fichiers a bien été réduit. On peut alors diminuer le volume logique en toute sécurité, jusqu'à une taille supérieure à 1,8 Go, puis demander au système de fichiers de prendre à nouveau toute la place :

Terminal

```
$ sudo lvreduce -L -700M /dev/mon_vg/super_lv
WARNING: Reducing active and open logical volume to 3,32 GiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce super_lv? [y/n]: y
 Reducing logical volume super_lv to 3,32 GiB
 Logical volume super_lv successfully resized
$ sudo resize2fs /dev/mon_vg/super_lv
resize2fs 1.42 (29-Nov-2011)
```

Terminal

```
Le système de fichiers de /dev/mon_vg/super_lv est monté sur /mnt ; le changement de
taille doit être effectué en ligne
old_desc_blocks = 1, new_desc_blocks = 1
Le système de fichiers /dev/mon_vg/super_lv a maintenant une taille de 869376 blocs.
$ df -h /mnt
Sys. de fichiers            Taille Utilisé Dispo Uti% Monté sur
/dev/mapper/mon_vg-super_lv   3,3G    5,3M  3,1G   1% /mnt
```

Notons cependant qu'il est parfois plus efficace et plus rapide de créer un autre volume logique, plus petit, puis de transférer les données de l'un à l'autre...

3.9 Supprimer un LV

Il arrive qu'un LV devienne inutile, pour diverses raisons. La commande à utiliser dans ce cas est **lvremove**. Il faut bien sûr l'avoir démonté auparavant :

Terminal

```
$ sudo umount /mnt
$ sudo lvremove /dev/mon_vg/super_lv
Do you really want to remove active logical volume super_lv? [y/n]: y
  Logical volume "super_lv" successfully removed
```

Avec les commandes **lvs** et **vgs**, on constate que le volume a bien été effacé et que de la place a été libérée :

Terminal

```
$ sudo lvs
LV       VG     Attr   LSize Origin Snap%  Move Log Copy%  Convert
autre_lv mon_vg -wi-a- 5,00g
$ sudo vgs
VG     #PV #LV #SN Attr   VSize  VFree
mon_vg   2   1   0 wz--n- 11,23g 6,23g
```

3.10 Agrandir un VG avec un PV

Lorsque l'espace disque dans un VG devient insuffisant, on peut très simplement y associer un nouveau PV. Reprenons le volume physique que nous avons créé plus haut sur **/dev/sdc1**. Pour ajouter celui-ci au groupe de volumes, la commande à utiliser est **vgextend**, celle-ci étant plutôt simple dans sa syntaxe, comme toutes les commandes rencontrées jusqu'ici :

Terminal

```
$ sudo vgextend mon_vg /dev/sdc1
Volume group "mon_vg" successfully extended
$ sudo vgs
VG     #PV #LV #SN Attr   VSize  VFree
mon_vg   3   1   0 wz--n- 19,22g 14,22g
```

On constate que le groupe de volumes fait maintenant un total de presque 20 Go, répartis sur nos trois disques !

3.11 Retirer un PV d'un VG

Parfois, on veut intégrer un nouveau disque dans un VG pour ensuite supprimer un ancien disque, par exemple pour exploiter des disques plus grands ou plus performants. Pour enlever un volume physique d'un groupe de volumes, la commande à utiliser est **vgreduce**. N'oublions pas que l'on a déjà des volumes logiques dans le groupe de volumes : il faut alors que le PV supprimé soit plus petit que l'espace libre dans le VG !

Vérifions donc d'abord avec les commandes **vgs** et **pvs** l'espace disponible dans le VG et les tailles des volumes physiques :

Terminal

```
$ sudo vgs
VG     #PV #LV #SN Attr   VSize  VFree
mon_vg   3   1   0 wz--n- 19,22g 14,22g
$ sudo pvs
PV         VG     Fmt  Attr PSize PFree
/dev/sda2  mon_vg lvm2 a-   3,23g 3,09g
/dev/sdb1  mon_vg lvm2 a-   8,00g 3,14g
/dev/sdc1  mon_vg lvm2 a-   8,00g 8,00g
```

Ici, on a 14 Go de libres et les plus grands volumes physiques font 8 Go : aucun problème, on peut supprimer celui que l'on veut !

Pour supprimer un volume physique non utilisé, on exécute la commande de cette manière-là :

Terminal

```
$ sudo vgreduce mon_vg /dev/sdc1
Removed "/dev/sdc1" from volume group "mon_vg"
```

Mais si l'on veut supprimer un volume utilisé, cela ne fonctionne pas :

Terminal

```
$ sudo vgreduce mon_vg /dev/sda2
Physical volume "/dev/sda2" still in use
```

On utilise alors d'abord la commande **pvmove** pour déplacer les données sur un autre volume (cela peut prendre du temps, mais la machine reste totalement disponible pendant l'opération), on peut ensuite le supprimer :

Terminal

```
$ sudo pvmove /dev/sda2 /dev/sdb1
/dev/sda2: Moved: 21,6%
/dev/sda2: Moved: 100,0%
$ sudo vgreduce mon_vg /dev/sda2
Removed "/dev/sda2" from volume group "mon_vg"
```

4. INSTALLER GNU/LINUX AVEC LVM

Suivant la distribution que vous utilisez, le média d'installation peut proposer (ou non) d'utiliser LVM. Entre autres, Debian et Fedora proposent LVM dans l'installeur habituel, alors qu'Ubuntu ne le propose pas. Pour utiliser Ubuntu avec LVM, il faut l'installer avec le média dit « alternatif » (*alternate*), qui supporte LVM contrairement à la version « desktop ».

Dans tous les cas, il est toujours possible après coup de mettre en place LVM sur le système ; de plus, c'est un exercice très intéressant :

- Installez le support de LVM dans la distribution ;
- Démarrez sur un *live CD* quelconque, tant qu'il supporte LVM ;
- Libérez un peu de place sur le disque dur (en réduisant ou supprimant une partition physique par exemple) ;
- Créez un PV, un VG, des LV sur votre disque ;
- Déplacez les données vers ces volumes, en agrandissant le VG progressivement si nécessaire ;
- Modifiez le fichier **etc/fstab** du système cible, afin de correspondre aux nouveaux volumes ;

- Utilisez **chroot** pour vous placer dans le système cible ;
- Mettez à jour le chargeur de démarrage, afin qu'il prenne en compte la nouvelle disposition.

Bien sûr, cette manipulation n'est pas des plus évidentes, n'hésitez pas à vous armer de patience et à vous documenter pour la réaliser !

Un schéma de partitionnement utilisant LVM pourrait être le suivant :

- une seule partition sur le disque dur ;
- un PV dans cette partition ;
- un LV de 20 Go, monté sur **/** ;
- un LV de 5 Go, monté sur **/home** ;
- un LV de 1 Go, monté comme swap.

De cette manière, vous avez la possibilité de créer d'autres LV selon vos besoins ou d'augmenter les LV existants, en utilisant tout l'espace que vous aurez laissé libre sur le disque.

Attention, selon le chargeur de démarrage, il peut être nécessaire de conserver **/** ou **/boot** en dehors de LVM : cela fait plusieurs années que GRUB supporte LVM, mais ce n'est pas le seul chargeur de démarrage qui existe et, qui sait, peut-être travaillez-vous sur du matériel exotique...

5. POUR ALLER PLUS LOIN

Avant de chercher à aller plus loin, commençons par nettoyer notre système en supprimant les LV, les VG et les PV qui restent :

Terminal

```
$ sudo lvs
LV       VG     Attr   LSize Origin Snap%  Move Log Copy%  Convert
autre_lv mon_vg -wi-a- 5,00g
$ sudo vgs
VG     #PV #LV #SN Attr   VSize VFree
mon_vg   1   1   0 wz--n- 8,00g 3,00g
$ sudo lvremove /dev/mon_vg/autre_lv
Do you really want to remove active logical volume autre_lv? [y/n]: y
  Logical volume "autre_lv" successfully removed
$ sudo vgremove mon_vg
Volume group "mon_vg" successfully removed
$ sudo pvs
PV         VG   Fmt  Attr PSize PFree
/dev/sda2       lvm2 a-   3,23g 3,23g
/dev/sdb1       lvm2 a-   8,00g 8,00g
/dev/sdc1       lvm2 a-   8,00g 8,00g
$ sudo pvremove /dev/sda2
Labels on physical volume "/dev/sda2" successfully wiped
$ sudo pvremove /dev/sdb1
Labels on physical volume "/dev/sdb1" successfully wiped
$ sudo pvremove /dev/sdc1
Labels on physical volume "/dev/sdc1" successfully wiped
```

Et si vous avez utilisé des images de disques, n'oubliez pas de les dissocier avec la commande **losetup**, comme vu précédemment !

Comment peut-on aller plus loin ? Il y a deux aspects intéressants que l'on peut approfondir. D'une part, la commande **lvcreate** avec l'argument **-s** permet de créer des instantanés (*snapshots*) de volumes logiques existants : pratique pour avoir la possibilité de retrouver des fichiers comme ils étaient avant une manipulation hasardeuse. D'autre part, l'option **-m** (ou **--mirrors**) de cette même commande permet de créer des volumes logiques dupliqués sur deux volumes physiques au sein d'un même groupe de volumes, à l'image de ce que l'on peut faire avec le RAID 1. ■

4

4 NET & SÉCURITÉ

À découvrir dans cette partie...

OPENSSH, L'ACCÈS À DISTANCE EN TOUTE SÉCURITÉ

Utiliser un ordinateur en ligne de commandes c'est bien pratique, mais ça ne serait pas aussi puissant sans la possibilité d'accéder et d'administrer à distance différents serveurs (ou clients, d'ailleurs) sans bouger de son bureau habituel...

À l'heure actuelle, la totalité des distributions GNU/Linux fournissent aussi bien un serveur qu'un client SSH ; il s'agit bien souvent de la suite logicielle OpenSSH. Mais commençons par le commencement. De tout temps, le besoin de prendre le contrôle d'une machine à distance a été très important. Qui n'a jamais regretté d'avoir oublié ses fichiers sur un autre ordinateur, ou encore qui ne s'en est pas voulu de n'avoir pas tapé telle ou telle commande avant de quitter son bureau ? Autre cas de figure, quand on est administrateur système, on n'imagine pas une seule seconde se lever de son bureau toutes les cinq minutes pour faire une opération sur un serveur. Ne parlons pas des systèmes qui n'ont même pas d'écran ni de clavier (les systèmes embarqués par exemple)...

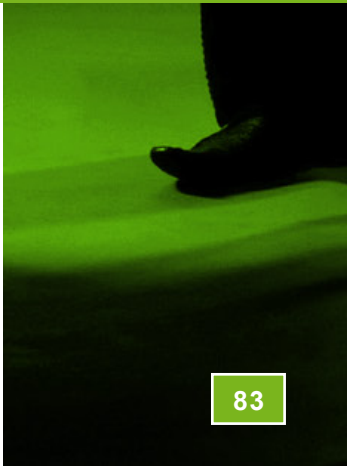
Pour ces raisons sont apparues les premières solutions de shells distants, comme `telnet` ou `rsh`. Le but est simple : obtenir un accès sur une machine pouvant se trouver n'importe où sur un réseau et pouvoir interagir avec le système comme si nous étions devant l'écran. Ces premières solutions étaient très simples, mais pourtant fonctionnelles. Elles avaient cependant un gros inconvénient : les échanges entre le poste client et la machine distante circulaient en clair sur le réseau ; au début, lorsqu'on faisait confiance aux autres utilisateurs, ce n'était pas un problème, mais avec la démocratisation des réseaux et d'Internet on ne trouve plus de réseau réellement sûr. Il est techniquement possible de voir et d'enregistrer tout ce qui se passe entre une machine et une autre au travers d'un réseau ; on appelle cela « sniffer » une connexion. Un individu peu scrupuleux peut ainsi tout savoir sur ce que vous faites : vos mots de passe, les commandes que vous utilisez, les résultats obtenus, etc. Bref, avec ces protocoles, il n'est vraiment pas difficile de pirater un système distant. La majorité des protocoles réseau fonctionnent d'ailleurs de cette manière : les données circulant sur un réseau ne sont généralement pas chiffrées.

Le protocole SSH a été inventé en 1995 par Tatu Ylönen, un informaticien finlandais, qui a ensuite créé la société *SSH Communications Security* pour exploiter cette invention commercialement. L'équipe du système d'exploitation libre OpenBSD a alors créé **OpenSSH** (première version parue en 1999), afin d'offrir une alternative libre au logiciel de cette société. SSH offre les mêmes fonctionnalités que `rsh` (connexion distante en ligne de commandes), mais les connexions sont chiffrées de bout en bout, les données ne circulent plus « en clair » sur le réseau.

De base, OpenSSH permet de remplacer les utilitaires fournissant une connectivité « shell » à distance. Mais nous verrons plus loin que de nombreuses autres fonctionnalités sont proposées, allant jusqu'à sécuriser n'importe quel protocole déjà existant. Il inclut également par défaut des fonctions de vérification des machines auxquelles on se connecte, pour être sûr qu'un attaquant n'intercepte pas la tentative de connexion afin de soutirer notre mot de passe.

À savoir

OpenSSH est entièrement développé comme un élément d'OpenBSD, sous la direction de Theo de Raadt. Cela n'empêche pas ce logiciel d'être portable, de pouvoir être installé sur de très nombreux systèmes informatiques, grâce à une équipe qui est dédiée à cette adaptation. Le code d'OpenSSH sur OpenBSD est donc bien plus léger que sur les autres systèmes.



1. INSTALLATION

OpenSSH se compose de deux éléments :

- le **client** OpenSSH, c'est le logiciel qu'on utilisera pour se connecter au travers du protocole SSH ;
- le **serveur** OpenSSH, c'est le programme qui attend des connexions afin de donner accès à la machine.

Dans la plupart des cas, le client OpenSSH est installé par défaut avec la distribution. Il n'en va pas toujours de même pour le serveur : la majorité des gens ne veulent pas d'un quelconque logiciel en écoute sur leur poste de travail, aussi sécurisé soit-il.

Dans le cas de Debian, Ubuntu et leurs dérivées, le serveur OpenSSH est proposé dans le paquet `openssh-server` :

Terminal
```
$ sudo apt-get install openssh-server
```

Vous remarquerez que des clés sont créées lors de l'installation du serveur OpenSSH, ce qui nous amène au chapitre suivant...

2. LES CLÉS SSH

Les clés SSH (de type RSA, DSA et ECDSA) servent à l'identification (dire qui on est) et à l'authentification (prouver qui on est) du serveur. Afin de bien comprendre, faisons un peu de théorie... Le protocole SSH repose sur un **chiffrement asymétrique** ; un tel chiffrement met en œuvre deux clés, une pour chiffrer et l'autre pour déchiffrer.

Pour prouver qui je suis, je produis une paire de clés. Je garde une clé secrète pour moi et je diffuse à qui veut bien la recevoir ma clé publique (non secrète par définition). Ainsi, si on me demande qui je suis, je peux le prouver de la manière suivante :

- on me donne une phrase à chiffrer, un challenge ;
- je procède au chiffrement et je renvoie le résultat ;
- la personne en face déchiffre le challenge avec ma clé publique ;
- s'il retrouve la même phrase après déchiffrement, cela prouve que je suis bien qui je dis être.

Tout cela sans partager une seule seconde une quelconque donnée secrète !

OpenSSH fonctionne sur le même principe. Lors de l'installation, des paires de clés sont créées et placées dans le répertoire `/etc/ssh` : `ssh_host_dsa_key`, `ssh_host_ecdsa_key` et `ssh_host_rsa_key` pour les clés privées et `ssh_host_dsa_key.pub`, `ssh_host_ecdsa_key.pub` et `ssh_host_rsa_key.pub` pour les clés publiques respectives.

Lorsque l'on se connecte pour la première fois à une machine avec la commande `ssh`, on ne connaît pas sa clé publique...

Terminal
```
client$ ssh 192.168.42.21
The authenticity of host '192.168.42.21 (192.168.42.21)' can't be established.
ECDSA key fingerprint is 19:b2:92:4e:da:f9:ba:c6:90:1a:20:31:5d:c3:5d:2b.
Are you sure you want to continue connecting (yes/no)?
```

SSH nous affiche alors l'**empreinte** (*fingerprint*) de la clé RSA du serveur, charge à nous de la vérifier : habituellement, c'est à l'administrateur de la machine distante de nous fournir cette information, par e-mail par exemple (cette donnée n'est pas secrète, il n'y a aucun risque à la partager... Par contre, assurez-vous que c'est bien l'administrateur en question qui vous a transmis l'information et non un pirate qui se fait passer pour lui). Il est courant, lors d'une première connexion, de ne pas vérifier cette information et de répondre *yes* ; c'est une mauvaise habitude, car lors de cette première connexion, nous n'avons aucune assurance que l'on se connecte à la bonne

machine. Si un pirate s'intercale entre vous et la machine dès votre première connexion, vous vous laisseriez alors berner.

Pour voir l'empreinte côté serveur, il faut utiliser la commande suivante :

Terminal

```
serveur$ ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 19:b2:92:4e:da:f9:ba:c6:90:1a:20:31:5d:c3:5d:2b  root@testssh (ECDSA)
```

Ici, on constate que la clé du serveur est bien celle que l'on reçoit lorsque l'on essaie de se connecter : on peut alors répondre `yes` sans crainte :

Terminal

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.42.21' (ECDSA) to the list of known hosts.
diamond@192.168.42.21's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)
[...]
```

Ici, une fois que le lien est établi, la connexion est chiffrée et le serveur SSH demande le mot de passe de l'utilisateur `diamond` (sans préciser de nom d'utilisateur à la commande `ssh`, le nom courant est utilisé). Rien n'est affiché lorsqu'on tape le mot de passe, c'est une mesure de sécurité : une personne regardant par-dessus votre épaule ne connaîtra même pas la longueur de votre mot de passe.

On peut essayer une seconde connexion, l'empreinte n'est alors plus présentée :

Terminal

```
client$ ssh 192.168.42.21
diamond@192.168.42.21's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)
[...]
```

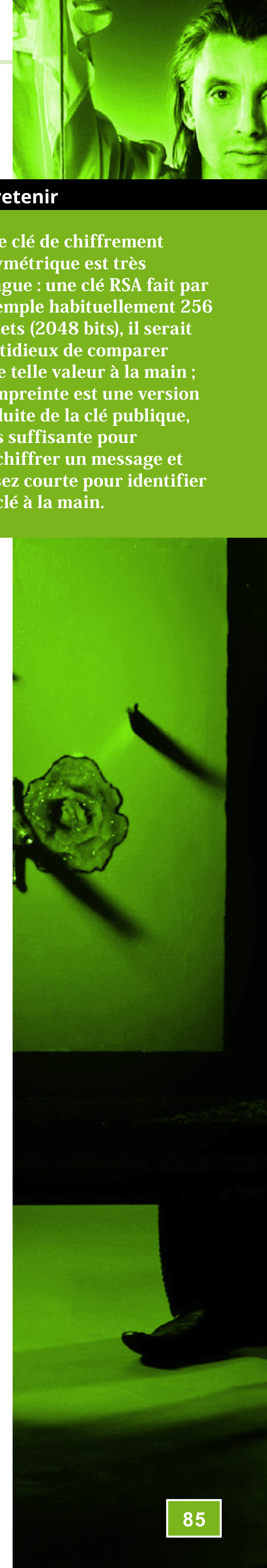
Le client SSH a lui-même vérifié l'identité du serveur, de manière automatique. Imaginons maintenant qu'un pirate se place entre nous et le serveur, qu'il détourne la tentative de connexion. Dans ce cas, il ne sera pas capable de chiffrer le *challenge* avec la clé privée et nous verrions immédiatement que ce n'est pas la bonne machine :

Terminal

```
client$ ssh 192.168.42.21
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
[...]
  remove with: ssh-keygen -f "/home/diamond/.ssh/known_hosts" -R 192.168.42.21
ECDSA host key for 192.168.42.21 has changed and you have requested strict checking.
Host key verification failed.
```

La connexion ne peut pas se faire, car l'identification de l'hôte distant n'est pas bonne. Peut-être que la clé privée du serveur a simplement changé : dans ce cas, il faut appeler l'administrateur de la machine pour valider la nouvelle clé. Si l'administrateur c'est vous, alors il est temps de vous poser des questions...

La commande `ssh-keygen` indiquée sur l'antépénultième ligne permet d'« oublier » la clé précédente et de recommencer comme si on n'avait jamais connu ce serveur.



À retenir

Une clé de chiffrement asymétrique est très longue : une clé RSA fait par exemple habituellement 256 octets (2048 bits), il serait fastidieux de comparer une telle valeur à la main ; l'empreinte est une version réduite de la clé publique, pas suffisante pour déchiffrer un message et assez courte pour identifier la clé à la main.

3. NOM ET PORT

On a vu comment se connecter avec le client OpenSSH : on utilise simplement la commande **ssh**. Bien sûr, celle-ci accepte différents paramètres, dont les plus courants sont **-l** pour se connecter sous un nom différent du nom courant, et **-p** pour se connecter à un port différent du port par défaut (port TCP 22). La commande suivante permet de se connecter sur le port 2222 en tant que **nimportequi** :

Terminal

```
$ ssh -l nimportequi -p 2222 192.168.42.21
```

Bien sûr, cette commande ne fonctionnerait pas, étant donné que le serveur SSH en question n'écoute pas sur le port 2222 et qu'il n'y a pas d'utilisateur appelé **nimportequi** sur la machine.

Une autre syntaxe pour choisir un nom d'utilisateur différent est la suivante : **<nom>@<adresse>**, ce qui pourrait donner :

Terminal

```
$ ssh -p 2222 nimportequi@192.168.42.21
```

La possibilité de choisir un nom d'utilisateur différent est facilement compréhensible : on n'a pas souvent, sur une machine différente, exactement le même nom d'utilisateur. Concernant le numéro de port, on peut par exemple faire des redirections de ports (NAT) d'une machine vers d'autres, donnant accès par une seule adresse IP à différents serveurs : on les différencie alors d'après le port utilisé.

4. AUTHENTIFICATION DES UTILISATEURS

En plus de fournir une connexion chiffrée entre deux machines, OpenSSH permet de déjouer des tentatives de piratage. Cela peut sembler inutile au premier abord, mais faire croire à un utilisateur qu'il se connecte à une machine sûre est une bonne méthode pour lui soutirer son mot de passe ; cela s'apparente à une attaque de type « man-in-the-middle ».

Il reste un point de fragilité dans le fonctionnement décrit jusqu'ici : la sécurité de l'accès ne repose que sur une seule chose : le mot de passe de l'utilisateur. Une bonne vue et un peu de baratin permet à beaucoup d'obtenir votre mot de passe, par exemple de vous voir le taper sur votre clavier. Impossible ? Utilisez-vous vraiment un mot de passe différent sur chaque service où vous vous connectez ? Vous ne tapez jamais un mot de passe sur votre smartphone dans un train, ou sur un PC public dans un cybercafé ? Si le mot de passe de votre messagerie est le même que sur votre serveur, alors il devient bien plus facile de l'obtenir : vous êtes certainement moins vigilant en lisant vos e-mails d'un cybercafé sur votre lieu de vacances qu'en vous connectant à votre serveur de chez vous...

Et même si on ne vous soutire pas le mot de passe, n'oubliez pas que leur fragilité est une problématique courante et très souvent évoquée !

À retenir

Certains changent systématiquement le port d'écoute d'OpenSSH, affirmant que cela apporte une sécurité complémentaire. En effet, sans changer le port, on voit dans les logs de nombreuses tentatives de piratage par des « script kiddies », qui tentent des noms d'utilisateurs et des mots de passe « standards », sur d'immenses plages d'adresses, en espérant obtenir un accès à l'une ou l'autre des machines testées.

Quand on change le port SSH, ces pseudo-attaques ne sont plus visibles dans les logs, vu que cette technique du « ratisser large » ne s'intéresse qu'au port 22. Mais cela n'apporte pas une sécurité complémentaire : tant que l'authentification des utilisateurs est assez forte, de telles tentatives ne peuvent qu'échouer. À la limite, cela apporte un certain confort pour ceux qui s'amusent à lire les logs d'authentification à longueur de journée...

Le cerveau humain a la fâcheuse tendance à être fainéant et un mot de passe a souvent une certaine logique, liée à la personnalité de son créateur : on retrouve la logique, on retrouve le mot de passe.

Pour pallier cette faiblesse, OpenSSH permet d'utiliser le système d'authentification par clé asymétrique pour valider l'identité du client qui se connecte. Ainsi, vous pouvez fournir une clé privée (qui est, par définition, très complexe) lors de la connexion au lieu d'un mot de passe. De plus, une clé privée d'utilisateur peut être protégée par une phrase de passe ; il est bien sûr conseillé de mettre en place une telle protection.

On parle bien de « phrase de passe » : une vraie phrase, avec des mots, des espaces, de la ponctuation... C'est bien plus sécurisé qu'un petit mot de quelques lettres et souvent plus facile à retenir ! Attention : cette phrase de passe n'est pas suffisante pour se connecter au serveur, elle ne sert qu'à déverrouiller la clé privée. Par conséquent, tant que vous gardez la clé privée secrète, il n'y a aucun risque. Et si vous avez des doutes quant à des accès frauduleux à cette clé privée, vous avez le temps de supprimer votre clé publique des serveurs et d'en créer une nouvelle avant que l'éventuel pirate ne devine votre phrase de passe.

Pour commencer, sur le client SSH, créons une paire de clés :

Terminal

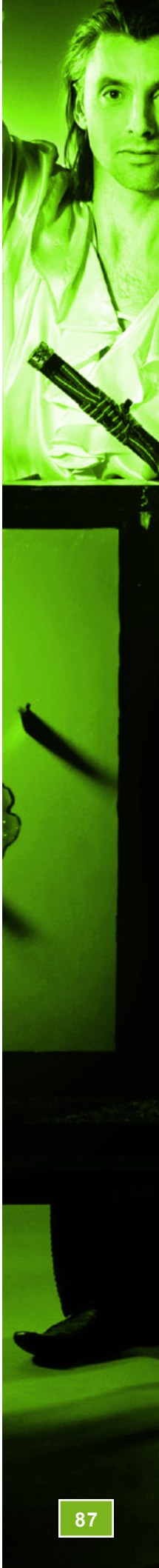
```
client$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/diamond/.ssh/id_rsa):
Created directory '/home/diamond/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/diamond/.ssh/id_rsa.
Your public key has been saved in /home/diamond/.ssh/id_rsa.pub.
The key fingerprint is:
8e:f2:03:97:31:a6:f5:ae:3a:1a:f8:89:b4:7f:63:6f diamond@client
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|      =          |
|     + =S        |
| .  o oo.        |
|... .o...        |
|.o.o.*.E.        |
|..=o+o*+         |
+-----------------+
```

On confirme l'emplacement de la clé à créer (le répertoire caché `.ssh` dans notre dossier utilisateur) et on indique (à deux reprises, comme d'habitude) la phrase de passe : celle-ci n'a rien à voir avec le mot de passe de votre compte utilisateur, vous y mettez absolument ce que vous voulez. OpenSSH affiche alors l'empreinte de la clé, ainsi qu'une image ASCII basée sur celle-ci : il est difficile de retenir une suite de caractères aléatoires, mais il est bien plus facile de reconnaître une image, même si elle est aléatoire elle aussi.

Une fois cette clé créée, on utilise la commande `ssh-copy-id` pour la placer sur le serveur :

Terminal

```
client$ ssh-copy-id 192.168.42.21
diamond@192.168.42.21's password:
Now try logging into the machine, with "ssh '192.168.42.21'", and check in:
  ~/.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```



Le mot de passe est demandé (pour la dernière fois), la clé publique est placée sur le serveur, puis cet outil recommande de vérifier les clés autorisées sur le serveur.

En effet, toutes les clés publiques utilisables pour se connecter à un utilisateur donné sont placées dans le fichier `.ssh/authorized_keys` de son dossier utilisateur. En consultant ce fichier, vous pouvez valider qu'il ne contient que la vôtre :

Terminal
```
serveur$ cat .ssh/authorized_keys
ssh-rsa AAA[...]ED5 diamond@client
```

À partir de maintenant, les connexions au serveur utiliseront la clé SSH et c'est sa phrase de passe qui sera demandée :

Terminal
```
client$ ssh 192.168.42.21
Enter passphrase for key '/home/diamond/.ssh/id_rsa':
```

En réalité, la commande `ssh-copy-id` ne fait que placer la ligne vue ci-dessus dans le fichier `.ssh/auhorized_keys` sur le serveur. Cette ligne est également présente sur le client, c'est la clé publique, dans le fichier `.ssh/id_rsa.pub` ; on peut alors la placer manuellement sur le serveur si on le souhaite.

Enfin, il peut être énervant de devoir taper une phrase de passe à chaque fois que l'on se connecte à une machine avec une clé SSH. Pour cela, il existe l'**agent** SSH. Son objectif est de fournir au client SSH la réponse au challenge, après avoir une seule fois demandé la phrase de passe à l'utilisateur : tant qu'il ne ferme pas sa session, celui-ci n'a alors plus besoin d'entrer sa phrase de passe.

L'agent est automatiquement lancé dans les environnements de bureau récents (KDE, GNOME, etc.). De cette manière, dès que vous utilisez une clé SSH, sa phrase de passe est demandée par l'environnement ; elle peut même être mémorisée dans le trousseau de clés de votre environnement.

En ligne de commandes, lorsque l'on n'utilise pas un environnement de bureau qui lance automatiquement l'agent, on peut exécuter la commande suivante pour le mettre en œuvre :

Terminal
```
$ eval $(ssh-agent)
```

Ensuite, il faut utiliser la commande `ssh-add` pour fournir la phrase de passe à l'agent :

Terminal
```
client$ ssh-add
Enter passphrase for /home/diamond/.ssh/id_rsa:
Identity added: /home/diamond/.ssh/id_rsa (/home/diamond/.ssh/id_rsa)
```

Après cela, on peut se connecter au serveur distant sans aucune demande : l'utilisation de SSH devient alors extrêmement aisée.

Terminal
```
client$ ssh 192.168.42.21
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)
```

5. TRANSFERT DE FICHIERS

OpenSSH ne permet pas seulement d'obtenir un shell sécurisé sur une machine distante. Il permet également de transférer des fichiers d'une machine à l'autre au travers de cette même connexion sécurisée.

Un tel transfert se fait avec la commande **scp**, dont l'utilisation est très simple :

Terminal
```
client$ scp toto.txt 192.168.42.21:titi.txt
toto.txt                                      100%  111KB 111.2KB/s   00:00
```

Ici, on transfère le fichier **toto.txt** sur le serveur distant, dans un fichier nommé **titi.txt**. On constate alors, sur le serveur, que ce fichier a bien été transféré :

Terminal
```
serveur$ ls -l titi.txt
-rwxr-xr-x 1 diamond diamond 113848 avril  2 14:07 titi.txt
```

Le transfert fonctionne bien sûr dans les deux sens ; on peut télécharger un fichier de la machine distante en le plaçant en premier argument, préfixé du nom ou de l'adresse IP de la machine :

Terminal
```
client$ scp 192.168.42.21:titi.txt nouveau_toto.txt
titi.txt                                      100%  111KB 111.2KB/s   00:00
client$ ls -l *toto*
-rwxr-xr-x 1 diamond diamond 113848 avril  2 14:11 nouveau_toto.txt
-rwxr-xr-x 1 diamond diamond 113848 avril  2 14:07 toto.txt
```

6. X11 FORWARDING

Une fonctionnalité très intéressante est la redirection du protocole X. Le protocole X, c'est celui qui est utilisé pour afficher des fenêtres graphiques, dans un environnement graphique. Avec le *X11 forwarding*, on peut exécuter un programme graphique sur la machine distante en déportant son affichage sur le client local. Bien sûr, cela nécessite une bonne bande passante, les applications graphiques étant souvent très gourmandes. Cela peut dépanner lorsque l'on doit temporairement exécuter un tel programme.

Pour activer cette fonctionnalité, on ajoute l'argument **-X** à la commande **ssh** :

Terminal
```
$ ssh -X <adresse IP>
```

7. REDIRECTIONS DE PORTS

SSH propose deux options pour rediriger des ports TCP, soit de la machine locale vers une machine du réseau distant, soit de la machine distante vers une machine du réseau local. Les options à la commande **ssh** sont respectivement :

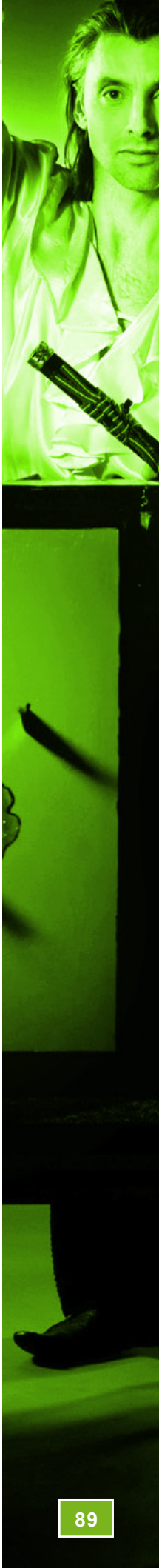
-L <port local>:<hote distant>:<port distant>

et :

-R <port distant>:<hote local>:<port local>

Imaginons, par exemple, que l'on se connecte sur une machine distante qui propose une interface web, mais que cette dernière n'est pas accessible par la connexion que l'on utilise (typiquement, une interface web non accessible par Internet). On peut alors utiliser la syntaxe suivante :

Terminal
```
$ ssh serveur -L 8080:localhost:80
```



Après s'être connecté au serveur en question, on peut se connecter sur le port 8080 local ; on se retrouve alors sur le site partagé par la machine « localhost » distante (port 80).

De même, dans l'autre sens, on peut permettre à une machine distante d'accéder à un port TCP quelconque du réseau local avec l'option `-R`.

Par ailleurs, on peut très bien mettre l'adresse d'une autre machine au lieu de « localhost » : on accède alors à un service qui est sur une machine sur laquelle on n'a même pas accès en SSH !

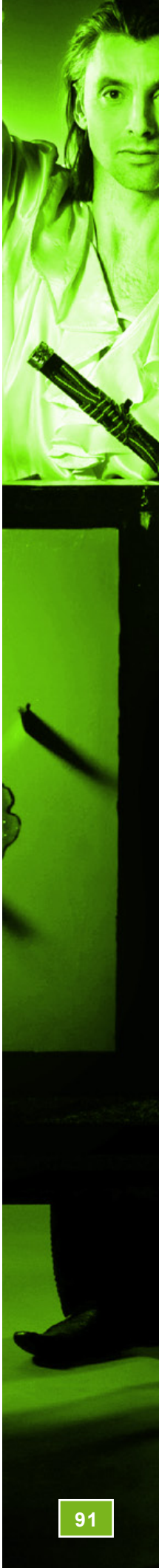
8. CONFIGURATION DU SERVEUR

La configuration globale d'OpenSSH est située dans le répertoire `/etc/ssh`. Celui-ci contient plusieurs fichiers :

- `moduli` : sans entrer dans les détails techniques et mathématiques, le protocole utilisé par OpenSSH repose sur des particularités concernant les nombres premiers (théorie des nombres) – ce fichier contient ce qu'on appelle des « moduli » (pluriel de modulo, le reste d'une division), utilisés pour la méthode d'échange de clés Diffie-Hellman ;
- `ssh_config` : c'est le fichier de configuration par défaut utilisé par le client OpenSSH ;
- `sshd_config` : celui-ci est le fichier de configuration du serveur OpenSSH lui-même ;
- `ssh_host_*_key` : comme nous l'avons vu plus tôt, ces trois fichiers sont les clés privées du serveur ;
- `ssh_host_*_key.pub` : comme ci-dessus, voici les trois clés publiques du serveur ;
- `ssh_import_id` : ce fichier configure l'outil `ssh-import-id`, que l'on peut utiliser pour récupérer des clés publiques à partir d'un serveur sur Internet – cet outil a été développé par Canonical, pour Ubuntu, il ne fait pas partie d'OpenSSH : en relisant le chapitre 1 de cet article, on retrouve le paquet correspondant à cet outil, installé en parallèle du serveur OpenSSH.

Nous venons de le voir, la configuration du serveur OpenSSH se trouve dans le fichier `/etc/ssh/sshd_config`. Celui-ci accepte de nombreux paramètres différents, en voici une liste non exhaustive...

- `Port <X>`, où `<X>` est un numéro de port : cette directive permet de définir sur quel port doit écouter le serveur OpenSSH ;
- `ListenAddress <adresse>` : cette fois-ci, on peut restreindre OpenSSH pour qu'il n'écoute que sur une interface réseau particulière (cette directive peut être répétée plusieurs fois) ;
- `Protocol <X>`, où `<X>` est 1 ou 2 : OpenSSH peut utiliser deux protocoles, SSH1 et SSH2 (le premier possède des vulnérabilités documentées et repose sur des algorithmes moins solides que le second). Dans la plupart des cas, on utilise le protocole SSH en version 2, sauf quand on doit permettre à des personnes ayant un client ne supportant que SSH version 1 et ne pouvant pas installer un autre client de se connecter – notons que le protocole SSH2 a été normalisé en 2006, ce n'est pas une nouveauté ;
- `HostKey <chemin>` : le chemin vers la clé privée du serveur – cette directive peut être utilisée plusieurs fois, notamment pour préciser les clés DSA, ECDSA et RSA ;
- `UsePrivilegeSeparation [yes|no]` : avec `yes`, le fonctionnement du serveur est sécurisé, une connexion entrante sera traitée par un processus n'ayant pas les droits du super-utilisateur root ;



- **SyslogFacility <facility>** et **LogLevel <level>** permettent de définir la façon dont les messages du serveur OpenSSH sont remontés dans le gestionnaire de journaux syslog ;
- **LoginGraceTime <secondes>** : ce paramètre définit le temps que le serveur OpenSSH laisse au client pour s'authentifier après avoir accepté une connexion – par défaut à 120, réduire cette valeur permet de limiter les blocages si un petit malin s'amuse à ouvrir de nombreuses connexions simultanées sur votre machine ;
- **PermitRootLogin [yes|no]** : avec ce paramètre, on autorise ou on interdit les connexions SSH directement avec l'utilisateur root – cela est bien sûr un risque de sécurité si on utilise un mot de passe faible, mais avec une bonne authentification (par clé notamment), cela ne pose pas de problème – cependant, lorsqu'une machine est utilisée par différentes personnes, il est préférable de les forcer à passer par leur compte habituel avant de « passer root » avec **su** ou **sudo**, pour savoir plus facilement « qui a fait quoi » ;
- **StrictMode [yes|no]** : cette option renforce encore la sécurité, car lorsqu'elle est à **yes**, OpenSSH refuse d'utiliser, pour l'authentification, des fichiers qui sont accessibles à d'autres utilisateurs que leur propriétaire, cela permet de limiter les corruptions qui pourraient être le fait d'autres utilisateurs de la même machine ;
- **PubkeyAuthentication [yes|no]** : ce paramètre active ou désactive l'authentification par clé ;
- **HostbasedAuthentication [yes|no]** : cette directive active ou désactive l'authentification par hôte – il ne s'agit plus de vérifier l'identité d'un utilisateur, mais simplement d'autoriser une connexion en se basant sur son adresse IP source – c'est bien sûr une approche très peu sécurisée, à éviter ;
- **PermitEmptyPassword [yes|no]** : on peut demander à OpenSSH d'autoriser les connexions sur des comptes dont le mot de passe est vide – c'est un risque de sécurité, le paramètre est à **no** par défaut, il vaut mieux l'y laisser ;
- **PasswordAuthentication [yes|no]** : comme pour les réglages ci-dessus, on peut activer ou désactiver l'authentification par mot de passe ;
- **KerberosAuthentication [yes|no]** : OpenSSH est capable de se baser sur Kerberos pour l'authentification – cette fonctionnalité mériterait un article entier à elle seule, nous n'allons pas épiloguer ici ;
- **X11Forwarding [yes|no]** : cette option permet d'activer ou désactiver la redirection des connexions au serveur graphique au travers de la connexion SSH ;
- **PrintMotd [yes|no]** : avec ce paramètre, on force (ou non) l'affichage du « message of the day » (fichier **/etc/motd**) lors de la connexion ;
- **PrintLastLog [yes|no]** : cette directive à **yes**, le serveur OpenSSH envoie au client, immédiatement après la connexion, l'horodatage de la précédente connexion – cela permet à l'utilisateur de vérifier que personne d'autre n'a utilisé son compte à distance ;
- **UsePAM [yes|no]** : OpenSSH se base par défaut sur PAM pour l'autorisation des utilisateurs, la vérification de leur authentification et la mise en place de leur session – on peut désactiver cela en mettant ce paramètre à **no** ;
- **Subsystem sftp /usr/lib/openssh/sftp-server** : cette option permet d'activer le sous-protocole SFTP, qui permet de parcourir le système à la manière d'un serveur FTP, mais de manière sécurisée.

D'autres paramètres existent, libre à vous de vous pencher sur la page de manuel **sshd_config**.

9. CONFIGURATION DU CLIENT

La configuration du client SSH est bien plus souvent personnalisée que la configuration du serveur. Lorsque l'on personnalise la configuration du client SSH, c'est en créant un fichier nommé **config** dans le répertoire **.ssh** du dossier personnel (ce qui donne **~/.ssh/config**). Celui-ci remplace alors le fichier par défaut, **/etc/ssh/ssh_config**.

Ce fichier est composé de différentes sections, titrées **Host <xxx>** où **<xxx>** est un identifiant d'hôte (c'est un nom plus ou moins libre). À partir de là, on peut définir différents paramètres par hôte, par exemple :

- **Hostname** : permet de définir le nom d'hôte réel auquel on se connecte lorsque l'on essaie de joindre l'hôte défini ;
- **User** : permet de préciser l'utilisateur sous lequel se connecter à la machine distante ;
- **Port** : le port auquel se connecter, si on n'utilise pas le port par défaut ;
- **ForwardX11** : active la redirection des requêtes graphiques (protocole X), sans avoir besoin de l'argument **-X** ;
- **Compression** : active la compression des données transférées, cela peut être très utile avec **ForwardX11** (mais ça consomme plus de processeur) ;
- **LocalForward** : équivalent de l'option **-L**, redirection d'un port local vers le réseau distant ;
- **RemoteForward** : équivalent de l'option **-R**, redirection d'un port distant vers le réseau local.

Le fichier **~/.ssh/config** peut par exemple ressembler à ça :

Fichier

```
Host serveur
    Hostname 192.0.2.1
    User root
    LocalForward 1143 localhost:143

Host pcperso
    Hostname 192.0.2.42
    User jeankevin
    Port 2222
    ForwardX11 yes
    Compression yes
```

Il n'y a alors plus qu'à utiliser le nom d'hôte défini à la main pour se connecter à la machine :

Terminal

```
$ ssh serveur
```

Et voilà, on est connecté au serveur et le port local 1143 est redirigé vers le port 143 distant. Sans ce fichier de configuration on aurait dû taper :

Terminal

```
$ ssh root@192.0.2.1 -L 1143:localhost:143
```

Pour la seconde entrée, on tape :

Terminal

```
$ ssh pcperso
```

au lieu de :

Terminal

```
$ ssh jeankevin@192.0.2.42 -p 2222 -X -C
```

10. CONNEXION PAR REBOND

Pour finir, découvrons une fonctionnalité particulièrement intéressante de SSH : la connexion par rebond.

Imaginons que l'on est sur une machine *alice* (192.0.2.10) et que l'on veut accéder à une machine *claire* (198.51.100.55), celle-ci n'étant joignable que par la machine *bob* (192.0.2.254). En utilisant simplement SSH comme on l'a appris, on ferait la chose suivante :

Terminal

```
alice$ ssh 192.0.2.254
bob$ ssh 198.51.100.55
claire$
```

Mais d'une part cela est moins pratique (il faut gérer l'authentification d'*alice* vers *bob*, puis de *bob* vers *claire*), d'autre part il n'est pas possible de transférer des fichiers d'*alice* à *claire*, ni de faire passer un flux X11 de *claire* vers *alice*.

L'idéal, ce serait qu'on puisse demander à *bob* d'ouvrir un tunnel vers *claire* pour emprunter ce tunnel afin d'accéder directement d'*alice* à *claire*... Un peu comme si on faisait une redirection de port, mais de manière automatique... Ça tombe bien, OpenSSH propose une association de fonctionnalités pour faire exactement cela ! Il s'agit de **`ProxyCommand`** et **`-W`**.

Concrètement, on crée les entrées suivantes dans **`~/.ssh/config`** sur *alice* :

Fichier

```
Host claire
    ProxyCommand ssh 192.0.2.254 -W 198.51.100.55:22
```

À partir de là, lorsque l'on tape **`ssh claire`** sur *alice*, les choses suivantes se passeront :

1. La **`ProxyCommand`** sera exécutée en premier ;
2. Cette **`ProxyCommand`** étant avant tout une connexion SSH vers *bob*, *alice* se connecte à *bob* ;
3. L'option **`-W`** fait que l'entrée et la sortie standard de la **`ProxyCommand`** (donc l'entrée et la sortie de cette connexion SSH de *alice* vers *bob*) sont redirigées vers le port 22 de *claire* ;
4. *alice* se connecte en SSH au travers de cette entrée et cette sortie standard, donc *alice* se connecte sur *claire*.

On peut alors profiter de ce mécanisme avec toutes les fonctionnalités qu'on a découvertes jusqu'ici...

... pour exécuter des applications graphiques sur *claire* et les afficher sur *alice* :

Terminal

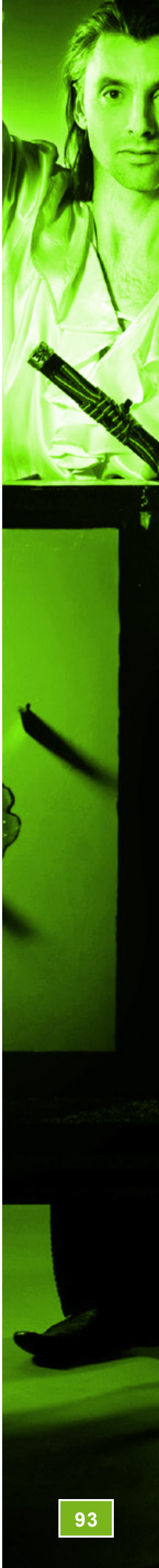
```
alice$ ssh -X claire
claire$ xcalc
```

... pour transférer des fichiers d'*alice* à *claire* :

Terminal

```
alice$ scp toto.txt claire:titi.txt
```

Enfin, on peut bien sûr intégrer l'option **`ProxyCommand`** aux côtés d'autres options d'OpenSSH, permettant par exemple avec **`ForwardX11`** de ne pas avoir besoin de l'argument **`-X`**. Par contre, l'option **`Hostname`** devient inutile : en effet, *alice* exécute la **`ProxyCommand`**, puis se connecte à l'entrée/sortie standard pour atteindre *claire*, pas une seule fois le nom d'hôte n'est nécessaire ! ▉



CONTRÔLER SON BUREAU À DISTANCE

Il existe différentes manières de rendre un environnement de bureau accessible de n'importe où, mais en général, ce sont des méthodes à mettre en place dès l'installation de l'ordinateur... Mais si on se rend compte, un peu tard, qu'on a oublié un fichier sur son ordinateur à 2h de route d'ici, que ce fichier est encore en cours d'édition sur LibreOffice et non sauvegardé, et qu'on en a besoin pour une réunion dans une demi-heure, une solution pour contrôler temporairement son bureau - histoire de sauvegarder ce fichier et de fermer LibreOffice - serait bienvenue !

Pour se sortir de cette situation, il faut bien sûr avoir accès d'une manière ou d'une autre à sa machine. Comme on est un geek un peu prévoyant, on a bien sûr mis en place un serveur OpenSSH sur son poste de travail et on a une redirection de port : ouf, on peut accéder à la machine avec SSH ! Mais cela n'est pas suffisant pour contrôler LibreOffice... C'est là que **x11vnc** entre en jeu !

Vous connaissez peut-être le protocole VNC : il permet de prendre le contrôle d'un ordinateur distant par le réseau. Par contre, il n'est pas sécurisé. Certes, il demande un mot de passe, mais l'ensemble des données (mot de passe compris) circulent « en clair », ce qui n'est pas souhaitable à travers Internet ; c'est pourquoi on l'associe à OpenSSH. **x11vnc**, de son côté, a l'avantage de rendre accessible par VNC une session graphique qui est déjà ouverte : d'autres logiciels permettent de mettre en place une seconde session, ce qui n'est pas notre objectif.

Dans cet article, nous appellerons la machine à laquelle on essaie d'accéder, **distant** et l'ordinateur sur lequel on est, **local**.

1. TUNNEL SSH

Tout d'abord, notons que le protocole VNC utilise par défaut le port 5900. Il faut alors rendre ce port, côté **distant**, accessible par le côté **local**. Pour cela, nous ferons une redirection de port avec l'option **-L** (ou **LocalForward**), évoquée dans l'article précédent. Par ailleurs, on va faire transiter des données graphiques, particulièrement gourmandes en bande passante ; on va alors compresser les données. Il y a deux possibilités pour ce faire :

- soit on exécute une commande une fois, occasionnellement, pour se sortir du pétrin :

Terminal
```
local$ ssh <adresse publique distante> -L 5900:localhost:5900 -C
```

- soit on est tête en l'air et on risque d'avoir besoin de cette commande souvent, auquel cas on met une section dans **~/.ssh/config** :

Fichier
```
Host monbureau
    Hostname <adresse publique distante>
    LocalForward 5900 localhost:5900
    Compression yes
```

... qu'on utilise alors de cette façon :

Terminal
```
local$ ssh monbureau
```

2. X11VNC

Il faut ensuite exécuter **x11vnc** sur la machine distante... D'abord, s'il n'est pas déjà installé, il faut bien sûr l'installer ; sous Debian, Ubuntu et leurs dérivées la commande serait :

Terminal
```
distant$ sudo apt-get install x11vnc
```

À savoir

Si vous voulez mettre en place un environnement graphique qui soit accessible à peu près tout le temps, x11vnc n'est pas la meilleure solution. On peut retenir en particulier x2go, qui utilise le protocole NX. Celui-ci est particulièrement adapté lorsque l'on souhaite utiliser un bureau à distance de manière continue ; la bande passante nécessaire est minimisée, la session est accessible à tout moment, SSH est utilisé pour chiffrer les communications...

On peut ensuite lancer **x11vnc**, en lui donnant les options adéquates :

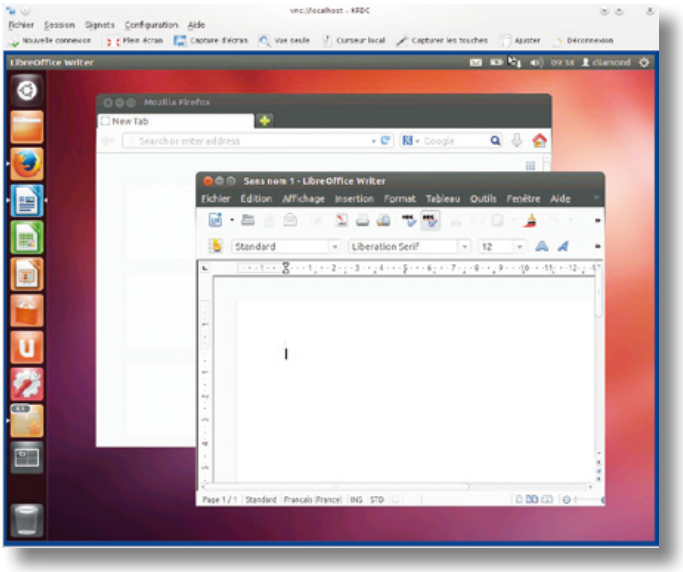
- ⇨ n'écouter que sur **localhost** (nous verrons pourquoi un peu plus loin) ;
- ⇨ partager la première session X11 (numérotée **:0**).

Terminal

```
distant$ x11vnc -localhost -display :0
[...]
The VNC desktop is:      localhost:0
PORT=5900
```

Parmi la myriade de messages retournés par **x11vnc**, on en relèvera deux :

- ⇨ Tout d'abord, **x11vnc** affiche une alerte indiquant que n'importe qui peut accéder à votre session, car il n'y a pas de mot de passe – dans notre cas, cela n'est pas un problème, grâce à l'écoute sur **localhost** : en effet, seul un utilisateur local peut alors utiliser cette session VNC ; à part vous, personne n'est actuellement connecté à votre ordinateur distant, n'est-ce pas ? On peut enlever ce message avec l'option **-nopw**.
- ⇨ Ensuite, on constate que **x11vnc** confirme la connexion à l'environnement **:0**, tout en indiquant que le port auquel se connecter est le port 5900.



Voilà, on peut se connecter avec un client VNC, comme **vncviewer**, Vinagre (sous GNOME) ou KRDC (sous KDE). Sur l'ordinateur local, on se connecte alors à l'adresse **localhost** : en effet, le port 5900 local est redirigé vers la machine distante, rappelez-vous !

Enfin, lorsque l'on ferme le client VNC, **x11vnc** s'arrête et rend la main, on peut alors se déconnecter avec **exit**.

3. TOUT FAIRE EN UNE FOIS

Si vous envisagez d'utiliser cette commande souvent, ça vous embêtera peut-être de taper systématiquement la commande **x11vnc** une fois connecté à distance... On va alors utiliser un alias pour simplifier tout ça.

Demandons d'abord à OpenSSH d'exécuter **x11vnc** dès qu'il est connecté :

Terminal

```
local$ ssh <adresse publique distante> -L 5900:localhost:5900 -C x11vnc
-localhost -display :0 -nopw
```

Créons alors un alias sur cette commande, en ajoutant une ligne dans le fichier **~/.bashrc** sur la machine **local** :

Fichier

```
# Alias pour accéder à mon bureau
alias monbureau='ssh <adresse publique distante> -L 5900:localhost:5900 -C
x11vnc -localhost -display :0 -nopw'
```

Fermons notre terminal, ouvrons-le à nouveau, puis exécutons simplement l'alias :

Terminal

```
$ monbureau
[...]
The VNC desktop is:      localhost:0
```

Difficile de faire plus simple, n'est-ce pas ? À la limite, avec une authentification par clé SSH sans phrase de passe et avec un lanceur placé sur le bureau de l'ordinateur local, vous pouvez ouvrir une connexion temporaire à la machine distante d'un simple double-clic...

4. QUELQUES EXPLICATIONS

La redirection de port mérite peut-être d'être un peu détaillée...

Lorsque l'on se connecte en SSH avec l'option **-L** (ou **LocalForward**), un port de la machine locale est redirigé vers la machine distante. Cela se fait de manière totalement transparente :

- Une application qui se connecte sur le port local croit réellement se connecter sur le port local, elle ne sait pas que le flux est redirigé ;
- L'application qui écoute sur le port distant ne sait pas que la requête vient d'ailleurs ; pour elle, cette requête vient de la machine distante.

Par conséquent, lorsque l'on se connecte au port 5900 à travers ce tunnel SSH, le serveur VNC voit la connexion venir de **localhost** : c'est pour cela qu'on demande à **x11vnc** de n'accepter que des connexions de la machine locale.

C'est également pour ça que l'alerte concernant le mot de passe n'est pas très importante ; vu que...

- **x11vnc** n'écoute que localement ;
- on n'a pas demandé à **x11vnc** d'accepter plusieurs clients ;
- on n'a pas demandé à **x11vnc** de rester en fonctionnement après la déconnexion ;

... alors le risque de sécurité ne se situe qu'entre le moment où on lance **x11vnc** et le moment où on s'y connecte :

- Si la connexion du client VNC est refusée, cela veut dire que quelqu'un s'est déjà connecté, quelqu'un qui a eu le temps de voir que vous avez lancé **x11vnc**, qui est connecté localement (soit sur la machine distante, soit sur la machine locale) et qui lance son client VNC plus vite que vous ;
- Si la connexion du client VNC est acceptée, alors toute connexion ultérieure sera refusée.

5. ALLER PLUS LOIN...

Ceci n'est en fait qu'une recette plutôt simple pour faire passer un flux VNC au travers d'une connexion SSH ; les tunnels SSH peuvent être utilisés dans de nombreuses situations, à vous d'imaginer les usages qui répondraient à vos besoins ! Concernant **x11vnc**, vous pouvez lire sa page de manuel, il offre de nombreuses autres fonctionnalités qui vous permettront de vous amuser de longues heures. Et si vous voulez mettre en place un bureau accessible à tout moment de n'importe où (un bureau « dans le cloud »), vous pouvez vous intéresser à d'autres outils, concurrents de **x11vnc**, peut-être plus adaptés à ce cas précis. ■

QUELQUES COMMANDES POUR « JOUER » AVEC TLS/SSL

SSL (ou plutôt TLS, c'est son nom depuis 2001, mais l'usage du terme SSL est resté) est un mécanisme de chiffrement de flux ; il est quotidiennement utilisé par tout internaute, au travers du protocole HTTPS. On peut faire énormément de choses avec SSL/TLS en ligne de commandes, un petit rappel des commandes utiles ne fait pas de mal...

Avec l'augmentation de la puissance de calcul et l'éveil des utilisateurs au sujet de la protection de la vie privée grâce à certaines affaires relativement médiatisées, l'authentification des tiers et le chiffrement des communications se fait de plus en plus présent. Par exemple, de plus en plus de services en ligne utilisent systématiquement HTTPS, même si c'est pour le simple affichage d'un panier d'achats.

En dehors du fait que cela consomme des ressources plus ou moins importantes, lorsque vous vous intéressez au fonctionnement de votre ordinateur cela nécessite de pouvoir manipuler les certificats SSL, les vérifier, les produire, etc. Pour cela, il existe des interfaces graphiques (comme TinyCA) et des outils complets de gestion de PKI.

Cependant, un outil en ligne de commandes vous permet la plupart des manipulations sans fioritures : **openssl**. Toute la difficulté réside dans les directives, options et arguments qu'il faut retenir. Voici donc une collection de commandes qui peuvent vous être utiles lorsque vous « jouez » avec OpenSSL, quelques recettes toutes faites, qu'il ne vous reste qu'à adapter à vos besoins !

1. AFFICHER « EN CLAIR » LE CONTENU DU CERTIFICAT D'UN SERVEUR

Terminal

```
$ echo "QUIT" | openssl s_client -connect mail.google.com:443 | openssl x509 -noout -text
```

Ici, on affiche le certificat de GMail, mais on peut bien sûr effectuer cela avec n'importe quel serveur qui implémente SSL, que ce soit par le protocole HTTPS ou un autre (IMAPS, SMTPS, etc.). Les options **-noout** et **-text** permettent respectivement de désactiver l'affichage du certificat encodé et d'afficher la version décodée. Vous pouvez spécifier d'autres options au lieu de **-text** :

- ⇨ **-subject** pour le sujet du certificat ;
- ⇨ **-serial** pour son numéro de série ;
- ⇨ **-dates** pour la validité du certificat.

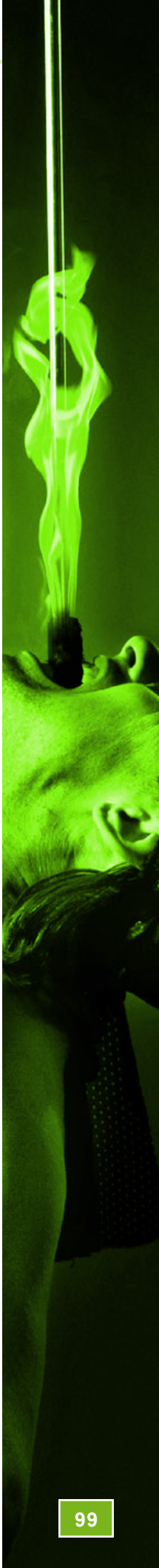
Bien entendu, ces options peuvent être utilisées en même temps. Sachez également que ceci fonctionne aussi bien pour une redirection que pour un fichier contenant un certificat, auquel cas on utilise l'option **-in**, suivie du chemin vers le fichier.

2. VÉRIFIER UN CERTIFICAT

Terminal

```
$ openssl verify -CAfile ca.pem certificat.pem
```

Ici, on fournit le certificat d'une autorité de certification ; on peut se passer de l'option **-CAfile** si on souhaite vérifier la validité d'un certificat en fonction des AC racines connues par le système, grâce à leurs certificats placés dans le répertoire **/etc/ssl/certs**.



3. CONVERSION DE FORMATS

➪ Pour convertir un fichier PKCS#12 au format PEM :

Terminal
```
$ openssl pkcs12 -in certificat.p12 -out certificat.pem -des
```

Lorsqu'on exporte un certificat (ou un couple certificat/clé) depuis Firefox par exemple, le format utilisé est généralement PKCS#12, alors que les outils en ligne de commandes préfèrent le format PEM. Notez l'argument **-des**, qui permet de traiter le chiffrement du même nom destiné à protéger la clé privée éventuellement intégrée au fichier.

➪ Pour convertir un fichier PEM au format PKCS#12 :

Terminal
```
$ openssl pkcs12 -export -inkey certificat.key.pem -in certificat.pem -out
certificat.p12
```

Le PKCS#12 a pour objectif de grouper la clé privée et le certificat, c'est pour cela qu'il faut deux fichiers entrants pour générer un fichier **.p12**.

➪ Pour convertir du format DER au format PEM :

Terminal
```
$ openssl x509 -in certificat.cer -inform DER -out certificat.pem
```

Comme il arrive parfois que les certificats d'AC de services en ligne soient en DER, il faut pouvoir faire la conversion. **-inform** est utile pour toute commande prenant un format non PEM en entrée.

4. GÉRER LE MOT DE PASSE D'UNE CLÉ

Ce n'est pas nécessairement une bonne idée, mais parfois on a besoin d'une clé privée sans mot de passe ; par exemple, dans le cas d'un serveur web HTTPS, on n'a pas envie de taper ce mot de passe à chacun de ses redémarrages (surtout qu'il y a toujours un risque de redémarrage quand on n'est pas derrière le clavier).
Pour supprimer le mot de passe d'une clé privée :

Terminal
```
$ openssl rsa -in certificat.key.pem -out certificat.key-nopass.pem
```

Pour changer le mot de passe d'une clé :

Terminal
```
$ openssl rsa -in certificat.key.pem -out certificat.key-newpass.pem -des
```

Notez que DES n'est qu'un algorithme de chiffrement symétrique parmi d'autres. De nos jours, on préférera **-aes256** ou **-camellia256**, si toutefois cela ne pose pas un problème de compatibilité avec le(s) logiciel(s) au(x)quel(s) on envisage de fournir cette clé.

5. TRANSFORMER UN CERTIFICAT EN DEMANDE DE CERTIFICAT

Terminal

```
$ openssl x509 -x509toreq -in certificat.pem -out certificat.csr -signkey
certificat.key.pem
```

Cela est parfois utile, mais c'est généralement une mauvaise idée. Techniquement, plus longtemps vous utilisez votre clé privée, plus vous avez de risques qu'elle soit compromise. Cependant, produire une demande de certificat (CSR) à partir d'un certificat déjà existant présente l'avantage de ne rien avoir à changer dans sa configuration. Vous gardez la même clé et, de ce fait, vous n'avez qu'à envoyer la CSR à l'autorité de certification pour avoir un nouveau certificat à jour. Mais un jour, la fainéantise vous perdra...

Pour ensuite afficher la demande de certificat, on utilisera presque la même commande que lors de l'affichage du contenu d'un certificat : `openssl req -noout -text -in certificat.csr`.

6. SIGNATURE DE FICHIER

Pour signer un fichier :

Terminal

```
$ openssl dgst -sha512 -sign certificat.key.pem -out monfichier.txt.sha
monfichier.txt
```

Cette commande a pour effet de produire un condensé (en anglais, *hash*) SHA512 du fichier `monfichier.txt`, puis de le chiffrer avec la clé privée contenue dans `certificat.key.pem` ; le résultat est alors placé dans le fichier `monfichier.txt.sha`. Cela permet d'assurer à un destinataire que le fichier `monfichier.txt` provient bien de vous...

Pour vérifier la signature d'un fichier :

Terminal

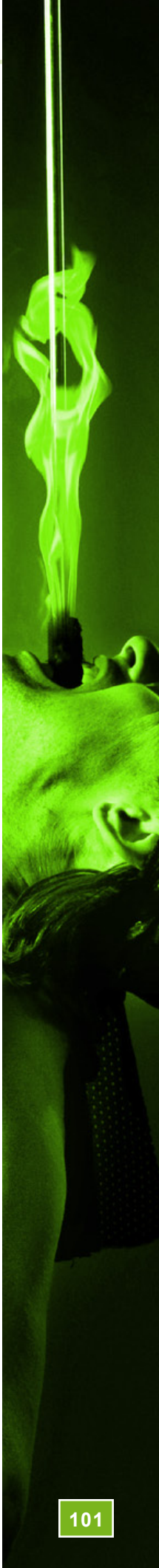
```
$ openssl dgst -sha512 -verify certificat.pub.pem -signature monfichier.
txt.sha monfichier.txt
```

Avec un fichier, sa signature et la clé publique de la personne qui a signé, on peut vérifier que le fichier provient bien de cette personne. Lorsque la signature correspond à la clé publique et au fichier, cette commande retourne `Verified OK`.

Mais avant de pouvoir vérifier la signature d'un fichier, il faut être en possession de la clé publique de la personne qui a créé cette signature. Elle est embarquée dans le certificat, mais celui-ci n'est pas directement utilisable. On produit donc un fichier indépendant qui contient uniquement la clé :

Terminal

```
$ openssl x509 -in certificat.pem -pubkey > certificat.pub.pem
```



7. (DÉ)CHIFFREMENT DE FICHIER

Terminal

```
$ openssl enc -aes-256-cbc -e -a -in fiche.pdf -out fiche.aes
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

Ici, on a chiffré le fichier `fiche.pdf` dans `fiche.aes` ; la commande a alors demandé le mot de passe de chiffrement. Les options utilisées sont bien sûr l'algorithme de chiffrement (`-aes-256-cbc`), `-e` pour demander de chiffrer (*encrypt*) et `-a` pour produire un fichier encodé en base64 (n'utilisant que des caractères de la table ASCII).

Si on regarde le type de fichier que cela donne, c'est assez simple :

Terminal

```
$ file fiche.aes
fiche.aes: ASCII text
```

On peut alors très simplement transmettre, embarquer voire, soyons fou, imprimer ce fichier.

Pour déchiffrer ce fichier, utilisons simplement `-d` (*decrypt*) au lieu de `-e` :

Terminal

```
$ openssl enc -aes-256-cbc -d -a -in fiche.aes -out quelquechose.bin
enter aes-256-cbc decryption password:
```

La commande demande bien sûr le mot de passe pour déchiffrer le fichier...

Enfin, si on vérifie le type du fichier résultat, c'est bien sûr le même qu'au début de la manipulation, un PDF :

Terminal

```
$ file quelquechose.bin
quelquechose.bin: PDF document, version 1.4
```

Bien sûr, AES n'est pas le seul algorithme de chiffrement supporté par OpenSSL. La commande suivante permet d'obtenir la liste des options disponibles, dont les algorithmes :

Terminal

```
$ openssl enc -h
```

En réalité, l'option `-h` n'existe pas ; et comme pour beaucoup de commandes, lorsque l'on demande une option inexistante, on reçoit la liste des options valides...

Pour coder/décoder en base64, on utilisera :

Terminal

```
$ openssl enc -base64 -e -in machin.odt -out machin.odt.b64
$ openssl enc -base64 -d -in machin.odt.b64 -out machin.odt
```

Le codage en base64 repose sur 64 caractères jugés standards, inclus dans la table ASCII et facilement imprimables (aucun caractère spécial : les 26 lettres majuscules et minuscules, les chiffres et les caractères `+` et `/`). Il permet de faire transiter une information sur n'importe quel type de média, ces caractères étant généralement supportés partout. Une donnée binaire est alors représentée sous forme d'un texte ASCII, il est plus facile de le traiter...

8. SIGNER EN S/MIME

Terminal

```
$ openssl smime -sign -in machin.odt.b64 -text -out mail.msg -signer
certificat.pem -inkey certificat.key.pem
```

S/MIME (pour *Secure Multipurpose Internet Mail Extensions*) est une norme et un standard de chiffrement et de signature électronique d'e-mails au format MIME. Il n'est normalement pas destiné à être utilisé en ligne de commandes, c'est au MUA (*Mail User Agent*, client de messagerie) de procéder au codage et au formatage des messages.

Il peut cependant être utile de savoir effectuer cette opération à la main avec **openssl** en cas de problème, ou alors pour traiter des données brutes en provenance de boîtes de messagerie ou de fichiers aux formats mbox ou Maildir.

Parmi les options utilisées, on remarque **-text**, qui n'a clairement pas la même signification qu'avec les autres commandes vues jusqu'ici : il ajoute les en-têtes MIME dans le fichier en sortie. Ici, nous avons choisi de signer le fichier, mais le chiffrement se fait presque de la même manière : c'est simplement le certificat du destinataire (clé publique) qui devra être utilisé...

Relevons également qu'une telle signature ne peut se faire que sur des fichiers textes (les e-mails étant par définition dans un format « texte brut »), voilà déjà un intérêt du codage en base64 des fichiers binaires.

Pour vérifier une signature S/MIME, on exécutera :

Terminal

```
$ openssl smime -verify -in mail.msg -CAfile ca.pem -out quelquechose.txt
$ openssl enc -base64 -d -in quelquechose.txt -out quelquechose.bin
```

Ici, on vérifie la signature, puis on décode le fichier en base64. En réalité, la vérification porte sur bien plus que la signature : elle porte également sur le condensé, son chiffrement et permet de remonter la chaîne de certificats jusqu'à celui d'une autorité de certification de confiance. Ici, nous spécifions le certificat de l'autorité, puisqu'elle n'est pas dans la liste des AC racines connues, mais l'option **-CAfile** n'est pas obligatoire.

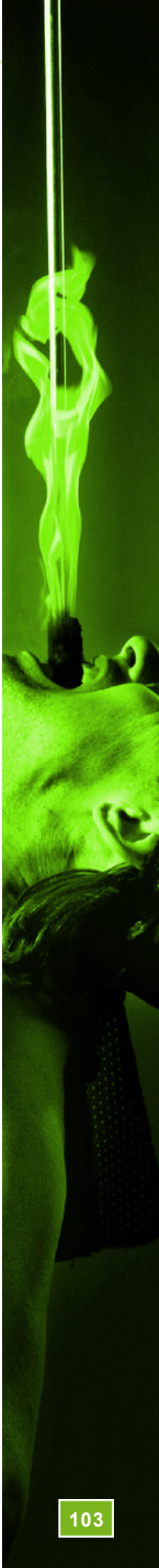
9. FAITES CHAUFFER VOS PROCESSEURS !

La commande **speed** d'OpenSSL permet de procéder à un ou plusieurs tests (*benchmark*) :

Terminal

```
$ openssl speed dsa -multi 15
```

Si vous ne spécifiez pas d'algorithme particulier, tous seront testés. Vous pouvez également en indiquer plusieurs en arguments derrière la commande **speed**. De la même manière que ce que nous avons vu plus haut, on peut utiliser **openssl speed -h** pour en avoir la liste. ■



5

5 MULTIMÉDIA

À découvrir dans cette partie...

5.1 ImageMagick, le traitement d'images en ligne de commandes

Référence du traitement d'images en ligne de commandes, ImageMagick permet toutes sortes de manipulations afin de transformer des images sans ouvrir une seule interface graphique. Pour le traitement par lots ou pour automatiser des transformations, c'est l'outil idéal. p. 106

5.2 Traitement vidéo et titrage avec Libav et ImageMagick

Tout comme les images, les vidéos peuvent être manipulées en ligne de commandes. Ce n'est pas parce qu'il s'agit d'images animées qu'on est obligé d'utiliser un système composé d'images animées pour les traiter. Libav est un outil très complet, qui permet de très nombreuses manipulations sur des vidéos sans ouvrir une seule fenêtre graphique. p. 118

IMAGEMAGICK, LE TRAITEMENT D'IMAGES EN LIGNE DE COMMANDES

ImageMagick est une boîte à outils rassemblant différentes commandes dédiées à la manipulation d'images. Grâce à cet ensemble d'outils, il offre des possibilités très étendues : conversion de format, redimensionnement, ajout d'effets graphiques ou d'éléments de décoration, retouche de photos, création de montages, ajout de texte, etc. Grâce à ces commandes, vous pourrez réaliser de nombreuses manipulations sans avoir besoin d'ouvrir des logiciels graphiques plus lourds... Et vous pourrez bien sûr automatiser ces manipulations si nécessaire.

Sur Debian et Ubuntu, ImageMagick est contenu dans le paquet nommé `imagemagick`, tout simplement...

1. DÉCOUVERTE D'IMAGEMAGICK

1.1 Contenu de la boîte à outils

ImageMagick se compose de onze commandes distinctes, chacune dédiée à un type d'opération :

- ⇨ `animate` permet d'animer à l'écran une série d'images ;
- ⇨ `compare` permet d'établir les différences, d'un point de vue mathématique et visuel, entre une image et sa reproduction (par exemple, entre une image avant et après compression JPEG) ;
- ⇨ `composite` permet de superposer deux images ;
- ⇨ `conjure` permet d'interpréter et d'exécuter des scripts écrits en MSL (*Magick Scripting Language*) ;
- ⇨ `convert` permet de réaliser des transformations sur une image (conversion de format, redimensionnement, découpage, effets graphiques...), l'image transformée étant stockée dans un autre fichier – c'est l'outil le plus utilisé et le plus connu parmi ceux proposés par ImageMagick ;
- ⇨ `display` permet d'afficher des images à l'écran ;
- ⇨ `identify` permet d'obtenir des détails sur une ou plusieurs image(s) ;
- ⇨ `import` permet de réaliser des captures d'écran (que ce soit l'écran entier, une fenêtre précise ou une portion quelconque de l'écran) ;
- ⇨ `mogrify` permet de réaliser des transformations sur une image, comme `convert`, mais écrase la source avec l'image transformée ;
- ⇨ `montage` permet de créer une image de type « mosaïque » à partir de plusieurs images ;
- ⇨ `stream` permet de traiter les pixels d'une image à la manière d'un flux de données, pixel par pixel.

Chacune de ces commandes propose de nombreuses options, l'ensemble couvrant ainsi l'éventail des manipulations que l'on peut avoir à effectuer sur des images.

1.2 Principe de fonctionnement

Avant d'aborder certains exemples, il est nécessaire de comprendre le mode de fonctionnement d'ImageMagick : ce logiciel stocke en mémoire une séquence d'images et travaille sur cette séquence d'images. Ensuite, sauf instruction contraire, il génère autant de fichiers cibles que d'images dans la séquence, en les numérotant.

On comprend donc bien que ce logiciel travaille sur des images, soit, mais il sait surtout travailler sur plusieurs images en même temps ; c'est sur ce principe que l'on basera certaines manipulations. Découvrons alors certaines options d'ImageMagick que nous aurons à utiliser plus loin :

- ⇨ `-clone X` permet de créer un clone d'une image de la séquence, `X` étant le numéro de l'image – si `X` est un nombre négatif, le compte commence à la fin ;
- ⇨ `+clone` est égal à `-clone -1` : la dernière image de la séquence est clonée ;
- ⇨ par défaut, ImageMagick travaille sur la séquence complète : pour créer un clone et l'insérer à la fin de la séquence, il faut mettre l'instruction `-clone` ou `+clone` entre parenthèses : toute instruction dans les mêmes parenthèses ne s'appliquera alors qu'à ce clone ;
- ⇨ `layers X` permet d'indiquer à ImageMagick de ne pas créer un fichier pour chaque image, mais de les combiner en un seul fichier, chaque image représentant alors un calque – `X` est alors la méthode utilisée pour combiner les images, par exemple `flatten` pour aplatir l'ensemble en

À savoir

Il est intéressant de noter l'existence d'un projet très similaire : GraphicsMagick. En réalité, celui-ci est un *fork* d'ImageMagick créé en 2002. Les développeurs de GraphicsMagick se sont concentrés sur l'optimisation et la stabilité du code, sans oublier bien sûr d'ajouter des fonctionnalités.
Les deux logiciels ont évolué indépendamment depuis 2002, mais ils restent très proches en termes d'utilisation : les commandes présentées ici sont utilisables avec GraphicsMagick. Par ailleurs, des *benchmarks* réalisés par l'équipe de GraphicsMagick indiquent des performances bien supérieures qu'avec ImageMagick. Enfin, notons qu'il est utilisé par de gros sites de traitement d'images, notamment Flickr.

Cet article se concentre sur ImageMagick, car c'est l'outil « historique », libre à vous d'expérimenter GraphicsMagick !

conservant la taille de la première image, ou `merge` pour aplatir l'ensemble, mais en élargissant le canevas suffisamment afin de contenir l'ensemble des images – vous pouvez consulter la liste complète des méthodes avec la commande `convert - list layers`.

À retenir

Pour l'interpréteur Bash (en réalité, pour l'écrasante majorité des shells), les parenthèses ont aussi une signification. Par conséquent, afin d'empêcher l'interpréteur de prendre en compte les parenthèses, il faut les échapper avec des anti-slashes, de la manière suivante : \([...] \).

2. TRANSFORMATIONS BASIQUES

ImageMagick est très utilisé pour effectuer les transformations basiques que l'on peut imaginer sur des images : format, redimensionnement, rotation, etc.

2.1 Changement de format

ImageMagick supporte de très nombreux formats d'image. La commande `identify -list format` permet de tous les lister ; sur la machine utilisée pour tester les commandes qui sont présentées dans cet article, 213 formats différents sont listés ! Grâce à ce support étendu de nombreux formats de fichiers, ImageMagick est parfaitement adapté pour convertir une image d'un format à un autre. Pour faire cela, deux outils sont à votre disposition : `convert` et `mogrify`.

Comme nous l'avons vu plus haut, `mogrify` édite un fichier sans en changer le nom ; quand on change le format de fichier, ce logiciel change également l'extension ! De son côté, `convert` nécessite qu'on lui précise le nom du fichier cible.

Pour convertir une image GIF en PNG, la commande à utiliser sera donc :

Terminal
```
~$ convert source.gif cible.png
```

ou bien :

Terminal
```
~$ mogrify -format png source.gif
```

Avec `mogrify`, on peut également effectuer une telle opération de transformation par lot, par exemple :

Terminal
```
~$ mogrify -format png *.jpg
```

Dans ce cas, on se retrouvera avec les équivalents `.png` de tous les fichiers `.jpg` présents dans le répertoire courant.

2.2 Redimensionnement

Un autre usage courant d'ImageMagick réside dans le redimensionnement d'image, avec l'option `-resize` de `convert` et `mogrify`.

Cette option prend un argument, qui peut avoir plusieurs formes :

- un pourcentage (par exemple `-resize 125%` ou `-resize 75%`) ;
- une nouvelle largeur en pixels (par exemple `-resize 640`) ;
- une nouvelle hauteur en pixels (par exemple `-resize x480`) ;
- une nouvelle largeur ET une nouvelle hauteur en pixels (par exemple `-resize 640x480`).

Dans le dernier cas, si le rapport entre largeur et hauteur n'est pas le même, l'image n'est pas étirée : la taille indiquée est une taille maximale : l'image résultante sera alors **au plus** à la taille demandée. Par exemple, une image de 200×100 redimensionnée à 700×200 aura en réalité une taille de 400×200. Le caractère utilisé pour séparer la largeur et la hauteur est la lettre minuscule « x ».

Pour réduire une image à 80% de sa taille, on utilisera alors la commande :

Terminal
```
~$ convert -resize 80% source.png cible.png
```

Pour obtenir une image dont la largeur serait de 480 pixels :

Terminal
```
~$ convert -resize x480 source.png cible.png
```

On peut également redimensionner toutes les images d'un répertoire :

Terminal
```
~$ mogrify -resize x480 *.jpg
```

Attention, cette dernière commande écrase les images sources par les cibles redimensionnées !

Notons également la syntaxe suivante, qui permet de ne redimensionner que les images plus grandes que la taille demandée : `-resize XXX>` où `xxx` est la taille cible.

Par exemple, la commande :

Terminal
```
~$ mogrify -resize x480> *.jpg
```

... ne redimensionnera que les images dont la hauteur est supérieure à 480 pixels, ignorant par exemple une image de 200×200.

2.3 Miniaturisation

On veut parfois générer des miniatures d'images, par exemple pour présenter une galerie de photos en ligne. Pour cela, l'option `-thumbnail` se montre plus adaptée que `-resize`, car elle procède à une série de transformations qui rendent cette miniaturisation plus rapide : c'est très utile lorsque l'on a des centaines d'images à générer.

Par ailleurs, `-thumbnail` enlève les métadonnées de l'image, afin d'obtenir un fichier le plus léger possible. On en profite pour découvrir l'option `-path`, qui permet d'indiquer à la commande que le résultat doit être placé dans un autre répertoire (cette option s'applique à n'importe quelle transformation, pas seulement à `-thumbnail`).

Terminal
```
~$ mogrify -path miniatures/ -thumbnail 128x128 *.jpg
```

Ici, toutes les images `.jpg` sont transformées en miniatures de 128×128 au maximum, qui sont placées dans le sous-répertoire `miniatures/` du répertoire courant.

L'argument de `-thumbnail` accepte les mêmes formes que pour `-resize`.

2.4 Inversion

Les options `-flip` et `-flop` permettent, respectivement, de retourner l'image selon un axe horizontal ou vertical. Autrement dit, `-flop` permet de faire un effet « miroir », tandis que `-flip` effectue un « renversement ».

Terminal
```
~$ convert source.jpg -flop cible.jpg
```

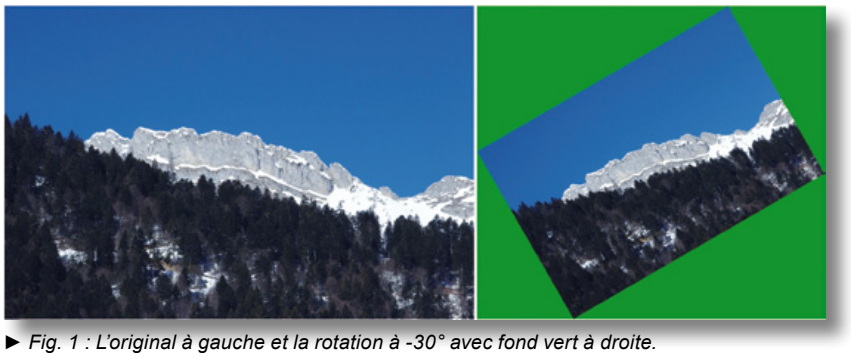

2.5 Rotation

L'option `-rotate` permet d'effectuer une rotation de l'image (Fig. 1). Elle prend comme argument l'angle de rotation en degrés ; elle accepte les valeurs négatives. Par exemple, pour effectuer une rotation de 90 degrés vers la gauche (dans le sens anti-horaire) :

Terminal
```
~$ convert source.png -rotate -90 cible.png
```

Les rotations avec des angles qui ne sont pas multiples de 90 font apparaître un fond blanc sur l'image. On peut changer l'image de fond avec l'option `-background`, par exemple :

Terminal
```
~$ convert source.png -background green -rotate -30 cible.png
```



► *Fig. 1 : L'original à gauche et la rotation à -30° avec fond vert à droite.*

Notons également la possibilité d'ajouter le caractère `>` ou `<` après l'angle de la rotation :

⇨ avec `>`, la rotation ne se fait que si la largeur est plus grande que la hauteur ;

⇨ avec `<`, la rotation ne se fait que si la largeur est plus petite que la hauteur.

Par exemple, la commande :

Terminal
```
~$ mogrify -rotate 90< *.png
```

... permet de « coucher » toutes les photographies verticales (portraits) et de conserver telles quelles les photographies horizontales (paysages).

3. ÉLÉMENTS DE DÉCORATION

ImageMagick permet non seulement de modifier des images existantes, mais aussi de créer des éléments graphiques, pour les intégrer à une image.

3.1 Cadres

Commençons par créer un cadre simple. Pour cela, on utilise l'option `-frame`, qui prend en argument l'épaisseur du cadre (épaisseur horizontale, puis verticale). On peut choisir la couleur du cadre avec l'option `-mattecolor`.

Terminal
```
~$ convert source.jpg -mattecolor orange -frame 20x20 cible.jpg
```

L'image sera alors agrandie d'autant, le cadre ne se superposant pas au contenu existant.

À savoir

Grâce à l'exemple de la rotation, on remarquera une particularité d'ImageMagick : les options sont appliquées les unes après les autres, dans l'ordre dans lequel on les ajoute. Si on avait placé l'option `-rotate` avant l'option `-background`, alors le fond de l'image aurait été blanc : le noir n'aurait été utilisé que pour les transformations suivantes, s'il y en avait eu.

Cela permet d'effectuer de nombreuses transformations sur une image, en une seule commande.

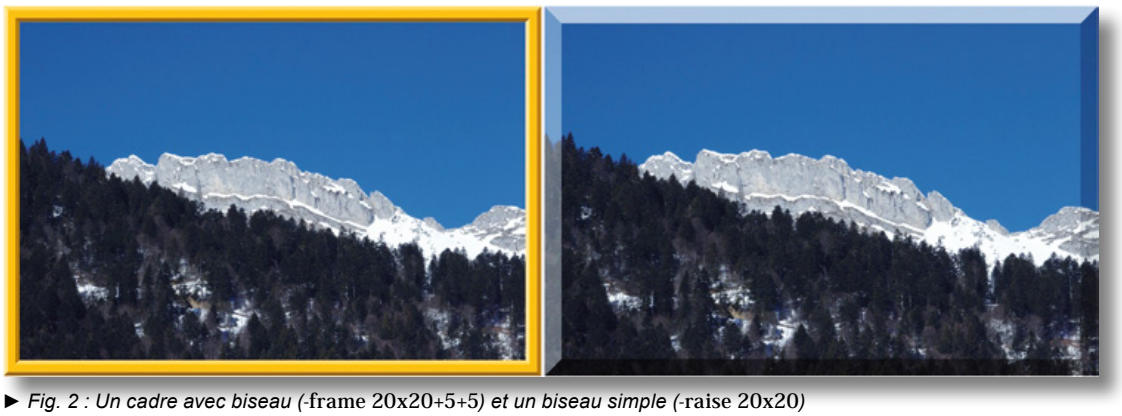
Pour un cadre biseauté (Fig. 2), qui sera donc visuellement plus sympathique, on précisera l'épaisseur du biseau interne, puis du biseau externe, avec le format indiqué dans cet exemple : `-frame 20x20+5+5`. Attention, la somme des biseaux ne peut pas être supérieure à la taille du cadre !

On peut également utiliser l'option `-raise` pour donner un effet « 3D » à l'image, avec un biseau sur les contours ; cette fois-ci, le biseau vient en transparence par-dessus l'image. Par exemple :

Terminal

```
~$ convert source.jpg -raise 20x20 cible.jpg
```

On peut aussi demander un biseau inversé (effet d'enfoncement dans l'écran), avec l'option `+raise` au lieu de `-raise`.



► *Fig. 2 : Un cadre avec biseau (*-frame 20x20+5+5*) et un biseau simple (*-raise 20x20*)*

On peut également ajouter une simple bordure avec l'option `-border` : la couleur doit alors être définie avec l'option `-bordercolor`. La commande suivante donne le même résultat que le premier exemple de `-frame` :

Terminal

```
~$ convert source.jpg -bordercolor orange -border 20x20 cible.jpg
```

Dans ce cas, l'option est beaucoup plus limitée, on ne pourra notamment pas appliquer de biseau à la bordure.

Notons que si on n'indique qu'un seul nombre (par exemple 20), alors la bordure est égale en largeur et en hauteur : `-border 20` est égal à `-border 20x20`.

3.2 Ombre portée

Pour créer une ombre, on utilise l'option `-shadow`. Celle-ci prend en argument une chaîne de la forme `AxB+C+D`, la seule valeur obligatoire étant `A` :

⇨ `A` : pourcentage d'opacité de l'ombre ;

⇨ `B` : le rayon du flou de l'ombre (sigma) ;

⇨ `C` : le décalage horizontal ;

⇨ `D` : le décalage vertical.

Si on ajoute `%` à la fin de l'argument, le décalage se fait en pourcents et non en pixels. Notons par ailleurs que cette ombre se fait sur un fond transparent : si on veut la visualiser, il faut alors utiliser un format de fichier supportant la transparence...

Mais cette option ne suffit pas à ajouter une ombre à une image : l'ombre est créée, mais elle reste désespérément seule dans son image. Il faut en réalité demander à `convert` de créer l'ombre, puis de l'appliquer sous l'image existante :

Terminal

```
~$ convert source.jpg \( +clone -background black -shadow 70x5+40+40 \)
+swap -background white -layers merge cible.jpg
```

On a ici une commande un peu plus complexe que les précédentes :

- on utilise `+clone` entre parenthèses pour, comme on l'a vu plus haut, travailler sur un clone de l'image ;
- on définit ensuite le fond de l'image en noir ;
- on crée alors une ombre, qui sera de la couleur du fond (noire), avec une opacité de 70%, des bordures floues sur 5 pixels et décalée de 40 pixels vers la droite et vers le bas ;
- on utilise ensuite `+swap` pour intervertir les deux dernières images de la séquence – en effet, lorsque l'on combine une séquence d'images, les dernières vont par-dessus les premières : pour que l'ombre se trouve *sous* la photographie, il faut alors qu'elle soit placée *avant* celle-ci dans la séquence ;
- ensuite, on définit la couleur de fond à blanc ;
- enfin, on demande de fusionner les images de la séquence dans un seul fichier, avec la méthode `merge` vue plus haut.

Notons que l'on a demandé un fond blanc pour l'ensemble de l'image, afin de pouvoir utiliser un format de fichier ne supportant pas la transparence (JPEG). Avec un format supportant la transparence (par exemple PNG), on aurait pu demander un fond transparent (`-background none`) ; l'image se fondrait alors parfaitement sur un site web n'ayant pas un fond uni par exemple.

► *Fig. 3 : Pour créer une ombre portée, on superpose une image et son clone flouté et coloré (ici en noir).*

3.3 Travailler avec du texte

Avant d'aborder le chapitre suivant, faisons une digression sur la création d'images basées sur du texte. En effet, ImageMagick est non seulement capable de travailler sur des images existantes et sur des groupes d'images existantes, mais il sait aussi générer des images.

Pour générer une image à partir d'un texte, on utilise la syntaxe de l'exemple suivant :

Terminal

```
~$ convert label:"Ceci est un texte" cible.png
```

Et si on veut choisir les propriétés de ce texte, différentes options sont disponibles, notamment :

- ⇨ **-background** pour choisir la couleur du fond ;
- ⇨ **-fill** pour choisir la couleur du texte ;
- ⇨ **-font** pour choisir la police – pour connaître les polices disponibles, la commande **convert -list font** est à utiliser ;
- ⇨ **-pointsize** pour définir la taille de la police.

On pourra par exemple utiliser la commande suivante :

Terminal

```
~$ convert -font Nimbus-Sans-Regular -pointsize 40 label:"Ceci est un texte" texte.jpg
```

On peut également utiliser le préfixe **caption** au lieu de **label** : dans ce cas, le texte trop long pour la largeur d'une image ira à la ligne si nécessaire.

3.4 Légendes

Il existe une option **-caption**. Les anglophones feront rapidement le lien avec le terme français *légende*, mais il ne faut pas se tromper : cette option ne permet pas d'insérer du texte dans l'image, elle ne fait que modifier la métadonnée « caption » de l'image, visible lorsque l'on demande à voir les propriétés détaillées de celle-ci dans un gestionnaire de fichiers…

On peut par contre combiner l'option **-polaroid** avec celle-ci : en effet, cette option utilise les métadonnées afin d'afficher du texte, en simulant un « Pola ».

Terminal

```
~$ convert -caption "Les Alpes à Albertville" source.jpg -polaroid 3 cible.jpg
```

Malheureusement, le résultat n'est pas du plus bel effet… N'abandonnons toutefois pas cet effet, qui a un bon rendu s'il est utilisé à bon escient : le faire fonctionner correctement sera un bon exercice !

► *Fig. 4 : L'effet Polaroid… un peu raté !*

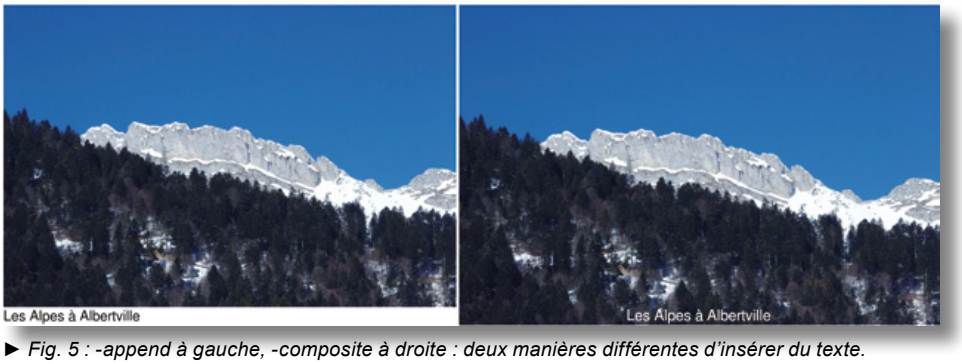
Une autre option est de créer une image contenant le texte en question et de la placer sous l'image sur laquelle on travaille. Reprenons l'exemple du chapitre précédent, puis utilisons la commande **-append** afin d'accoler les deux images de la séquence l'une à l'autre, de haut en bas (pour accoler de gauche à droite, la commande à utiliser est **+append**) :

Terminal
```
~$ convert montagne.jpg -font Nimbus-Sans-Regular -pointsize 40 label:"Les
Alpes à Albertville" -append append.jpg
```

Si on veut inclure le texte dans l'image (Fig. 5), on peut utiliser l'option **-composite**, qui crée un effet de composition entre deux images. À cette option, on associe **-gravity**, qui permet d'indiquer sur quel bord ou sur quel coin se fait l'association des images.

Terminal
```
~$ convert montagne.jpg -font Nimbus-Sans-Regular -pointsize 40 -background none
-fill white label:"Les Alpes à Albertville" -gravity south -composite composite.jpg
```

On aurait également pu utiliser l'option **-layers merge** au lieu de **-composite**, mais elle n'aurait pas respecté l'option **-gravity** : elle se base sur les coordonnées en termes de canevas pour placer les images les unes par rapport aux autres : on aurait dû d'abord agrandir le canevas du texte afin de le placer à l'endroit souhaité. Ceux qui essaient de comparer ces deux options se rendront alors compte que **-layers merge** donne le même résultat que **-gravity north-west -composite**.



► *Fig. 5 : -append à gauche, -composite à droite : deux manières différentes d'insérer du texte.*

3.5 Vignettage

Le vignettage, c'est l'effet que l'on peut apercevoir sur certaines photographies, mêmes récentes, où les coins de l'image sont plus sombres que le reste : l'explication physique de cet effet, dans les appareils photos, est liée à l'insuffisance de luminosité de l'objectif. Cet effet est parfois très important, les connaisseurs en photographie auront tout de suite à l'esprit le *Holga*... Les autres pourront chercher ce terme sur le Web !

Avec ImageMagick, on peut simuler cet effet avec l'option **-vignette**. Elle prend en argument une chaîne du type **AxB+C+D**, où seul A est obligatoire :

⇨ **A** : rayon du vignettage (en général on indique **0**) ;

⇨ **B** : étendue du flou (sigma) ;

⇨ **C** et **D** : espace entre le bord du vignettage et le bord de l'image.

Des valeurs positives de **C** et **D** induisent un vignettage plus petit que l'image ; des valeurs négatives induisent un vignettage plus grand (visible alors uniquement dans les angles). Par défaut, ces valeurs sont de 10%.

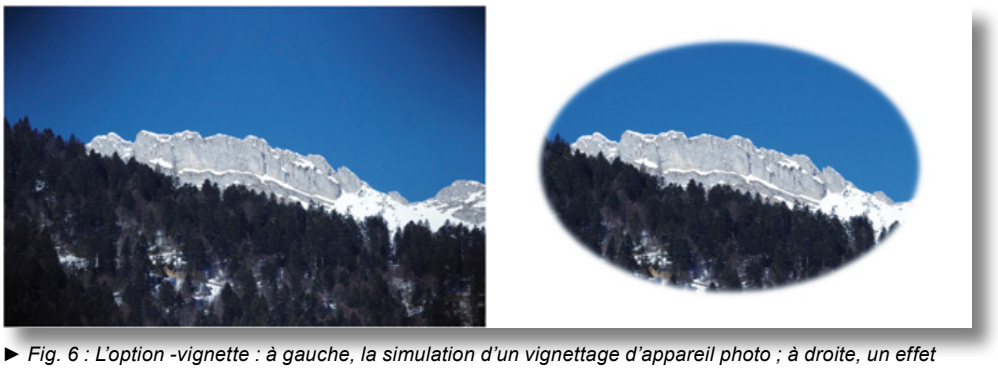
Sur notre photo d'exemple, la commande suivante permet de simuler un vignettage comme on pourrait en avoir avec certains matériels inadaptés :

Terminal
```
~$ convert source.jpg -background black -vignette 0x200-150%-150% cible.jpg
```

Le vignettage est un effet que certains photographes essaient d'exploiter, en donnant un aspect « vieilli » à leurs photographies.

On peut également utiliser cette option pour créer un médaillon d'une image :

Terminal
```
~$ convert source.jpg -vignette 0x10 cible.jpg
```



► *Fig. 6 : L'option -vignette : à gauche, la simulation d'un vignettage d'appareil photo ; à droite, un effet de médaillon.*

4. EFFETS ET RETOUCHE

ImageMagick permet d'effectuer certaines modifications sur une image, sur le même modèle que certains effets proposés par les logiciels de traitement d'image graphiques. Nous en voyons ici quelques exemples.

4.1 Peinture à l'huile

On peut simuler un effet « peinture à l'huile » avec l'option **-paint** d'ImageMagick. Celle-ci prend comme argument un seul nombre : le rayon (la taille) du pinceau que l'on simule :

Terminal
```
~$ convert source.jpg -paint 11 cible.jpg
```

4.2 Négatif

Le négatif est l'un des effets les plus courants. Celui-ci se fait avec l'option **-negate**. On peut également l'associer à l'option **-channel**, qui indique qu'il ne faut que travailler sur un canal particulier :

Terminal
```
~$ convert source.jpg -channel blue -negate cible.jpg
```

L'option **-channel** accepte les valeurs suivantes : **red**, **green**, **blue**, **alpha**, **cyan**, **magenta**, **yellow**, **black**, **opacity**, **index**, **rgb**, **rgba**, **cmyk** et **cmyka**.

4.3 Corriger une sous-exposition

Lorsqu'on prend une photo avec trop d'ombre, on obtient assez facilement une image « bouchée » : certaines parties semblent bien trop sombres. L'option **-sigmoidal-contrast** permet d'augmenter le contraste d'une photo sans pour autant saturer les zones de lumière. Elle prend deux arguments, séparés par un **x** :

- ⇨ l'intensité de la correction de contraste (cette valeur est généralement assez basse, essayez entre 1 et 4) ;
- ⇨ le réglage de la luminosité de 0 (vers le blanc) à 100% (vers le noir).

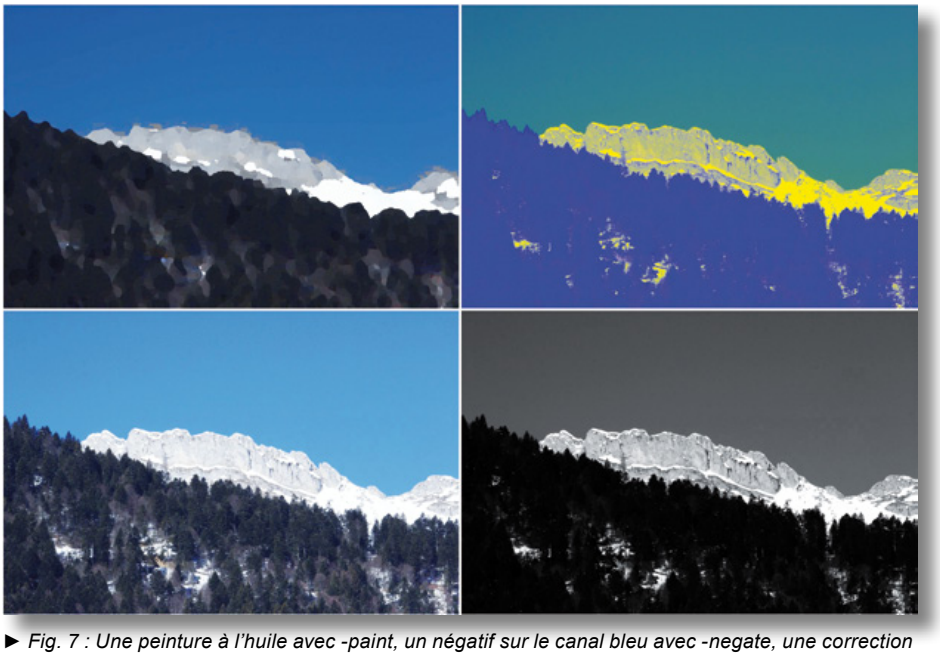
Terminal
```
~$ convert source.jpg -sigmoidal-contrast 4,20% cible.jpg
```

4.4 Noir et blanc

Avec l'option `-colorspace`, on peut changer l'espace de couleurs d'une image. On peut par exemple utiliser cette option pour transformer une image couleurs en noir et blanc, en demandant l'espace de couleurs gris :

Terminal

```
~$ convert source.jpg -colorspace gray cible.jpg
```



► *Fig. 7 : Une peinture à l'huile avec -paint, un négatif sur le canal bleu avec -negate, une correction de contraste avec -sigmoidal-contrast et une image monochrome avec -colorpsace gray.*

5. ASSEMBLAGE

5.1 Mosaïque avec convert

On a déjà vu comment assembler plusieurs images d'un séquence en les superposant, avec un éventuel décalage de l'une par rapport à l'autre. On peut utiliser la même approche pour assembler des images sous forme d'une mosaïque, en ajoutant simplement l'option `-page` : celle-ci permet d'indiquer où, dans le canevas global, s'intègre chaque image.

Par exemple, si on a quatre images `no.jpg`, `ne.jpg`, `so.jpg` et `se.jpg`, et qu'elles font toutes 640×480 pixels, on peut les arranger en mosaïque avec la commande suivante :

Terminal

```
~$ convert no.jpg -page +640+0 ne.jpg -page +0+480 so.jpg -page +640+480 se.jpg
-mosaic cible.jpg
```

Ici, nous avons utilisé l'option `-mosaic`, qui est simplement un raccourci vers `-layers mosaic` : on retrouve un nommage connu ! La différence entre `merge` et `mosaic` est assez mince : `merge` agrandit le canevas dans tous les sens afin d'inclure toutes les images, alors que `mosaic` n'agrandit que dans le sens positif (vers la droite et vers le bas) ; si une image est placée à une coordonnée négative, elle sera ignorée par `mosaic`.

5.2 Mosaïque avec montage

En réalité, pour assembler des images sous forme de mosaïques, la commande la plus adaptée est **montage**. Avec la commande suivante, on obtiendra le même résultat que ci-dessus :

Terminal
```
~$ montage no.jpg ne.jpg so.jpg se.jpg -mode Concatenate cible.jpg
```

Cette commande se charge toute seule d'organiser les fichiers correctement : il y en a quatre, elle décide alors de créer une image composée de deux lignes et deux colonnes. On peut forcer l'agencement des images avec l'option **-tile**. Par exemple, si on veut mettre les images sur une seule ligne :

Terminal
```
~$ montage no.jpg ne.jpg so.jpg se.jpg -mode Concatenate -tile x1 cible.jpg
```

Lorsque l'on veut laisser un espace entre les différentes images du montage, on peut utiliser l'option **-geometry** de la manière suivante :

Terminal
```
~$ montage no.jpg ne.jpg so.jpg se.jpg -geometry +4+4 cible.jpg
```

Un exemple de résultat ? Regardez simplement les images illustrant cet article : toutes les images groupées ont été réalisées avec cette commande !

Bien sûr, **montage** accepte de nombreuses autres options afin d'effectuer des montages qui peuvent devenir très complexes...

6. CAPTURE D'ÉCRAN

Intéressons-nous enfin à la commande **import**, qui permet d'effectuer des captures d'écran. Lorsqu'on l'exécute avec comme seul argument un nom de fichier, elle attend que l'on sélectionne :

- soit une fenêtre à prendre dans son ensemble (clic simple sur la fenêtre) ;
- soit une zone de l'écran à sélectionner par glisser-déposer.

Intéressons-nous à deux options de cette commande :

- **-pause X** permet d'attendre **X** secondes avant d'afficher le curseur de sélection de zone ;
- **-window X** permet de prendre automatiquement une capture de la fenêtre concernée – pour tout l'écran, on utilise alors le nom **root**.

La commande suivante attend 5 secondes et prend une capture de l'écran entier :

Terminal
```
~$ import -pause 5 -window root capture.png
```

Malheureusement, cette commande ne gère pas bien la composition, ni la transparence des fenêtres, et les effets d'ombres des environnements de bureau récents sont alors remplacés par un noir du plus mauvais effet...

7. ALLER PLUS LOIN

Vous en avez maintenant l'habitude, on n'a fait ici que survoler quelques fonctionnalités intéressantes d'ImageMagick. Son site officiel donne de très nombreux exemples et vous pourrez constater jusqu'où on peut aller avec ce logiciel : **http://www.imagemagick.org/Usage/**. ▮

TRAITEMENT VIDÉO ET TITRAGE AVEC LIBAV ET IMAGEMAGICK

Bien souvent, lorsque l'on veut manipuler des flux vidéo, on se sent « obligé » d'utiliser un logiciel graphique de montage vidéo. Mais on peut très bien faire ce genre de manipulations en ligne de commandes, soit pour automatiser un traitement, soit simplement parce qu'on est allergique à la manière dont fonctionnent ces logiciels. Voici un aperçu des manipulations que l'on peut envisager.

On n'est pas tous des fanas de montage vidéo, des spécialistes de l'image, des cadors d'Adobe Premiere (d'autant qu'Adobe n'est ni libre, ni compatible avec les systèmes sous Linux). Pourtant, on a parfois besoin de faire un peu plus que de simplement distribuer une vidéo qu'on a filmée en la transcodant dans un autre format : on peut notamment vouloir ajouter un titre ou un logo en surimpression (*overlay* ou *watermark*).

Plutôt que d'apprendre à utiliser une interface toute nouvelle (que ce soit un logiciel propriétaire ou libre) pour un besoin simple, pourquoi n'utiliserions-nous pas des outils en ligne de commandes, qui ne sont somme toute pas compliqués à utiliser ? En tout cas, il est bien plus simple d'expliquer leur fonctionnement en quelques pages de texte, là où certains logiciels graphiques ont besoin de livres entiers remplis de captures d'écrans.

On s'intéressera alors ici à deux outils très répandus : **ImageMagick** (que l'on a déjà abordé) et **Libav**, une boîte à outils complète pour le traitement vidéo – on trouvera ce dernier outil dans le paquet appelé simplement `libav-tools`, en tout cas sur Debian, Ubuntu et leurs dérivées. Dans le cadre de cet article, nous nous baserons sur une vidéo tournée avec un appareil photo Canon EOS 600D, c'est alors une vidéo au format MPEG4 dans un conteneur QuickTime, d'extension `.MOV`.

1. DÉCOUPER LA VIDÉO

La première manipulation qu'on souhaite généralement faire avec une vidéo, c'est la découper afin de ne conserver que la partie intéressante. Dans notre exemple, on souhaite conserver la partie qui commence à la 25e seconde et qui termine à 1 minute 38. On utilise alors la commande `avconv` :

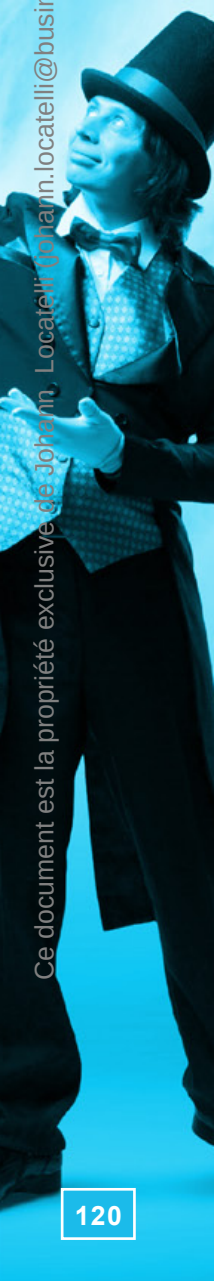
Terminal

```
$ avconv -i MVI_4764.MOV -ss 00:00:25 -t 00:01:13 -codec copy extrait.avi
avconv version 0.8.10-6:0.8.10-0ubuntu0.13.10.1, Copyright (c) 2000-2013
the Libav developers
  built on Feb  6 2014 20:53:28 with gcc 4.8.1
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'MVI_4764.MOV':
  Metadata:
    major_brand     : qt
    minor_version   : 537331968
    compatible_brands: qt  CAEP
    creation_time   : 2013-11-10 15:04:33
  Duration: 00:02:01.40, start: 0.000000, bitrate: 45836 kb/s
    Stream #0.0(eng): Video: h264 (Constrained Baseline), yuvj420p,
1920x1080, 44293 kb/s, 25 fps, 25 tbr, 25k tbn, 50k tbc
    Metadata:
      creation_time   : 2013-11-10 15:04:33
    Stream #0.1(eng): Audio: pcm_s16le, 48000 Hz, stereo, s16, 1536 kb/s
    Metadata:
      creation_time   : 2013-11-10 15:04:33
Output #0, avi, to 'extrait.avi':
  Metadata:
    major_brand     : qt
    minor_version   : 537331968
    compatible_brands: qt  CAEP
```

À savoir

Début 2011, une importante partie des développeurs mécontents de la manière dont les développements étaient intégrés dans FFmpeg ont décidé de « réorganiser » le projet, leur action ressemblant à un coup d'état : le mainteneur et contributeur principal (Michael Niedermayer) a été démis de ses fonctions, ses accès ont été retirés, etc. ; cela ne s'est pas très bien passé… Trois ans plus tard, la situation est un peu plus claire : étant donné qu'une partie de l'équipe de FFmpeg reconnaissait encore Niedermayer comme le responsable du projet, les développeurs « dissidents » ont décidé de créer un *fork*, nommé Libav.

Dans cet article, nous utilisons Libav simplement car c'est le logiciel proposé par défaut sur Ubuntu. Si vous voulez effectuer les mêmes manipulations avec le « vrai » FFmpeg, vous pourriez avoir besoin de faire des modifications dans les commandes et leurs arguments…



Terminal

```
    creation_time   : 2013-11-10 15:04:33
    ISFT            : Lavf53.21.1
    Stream #0.0(eng): Video: libx264, yuvj420p, 1920x1080, q=2-31, 44293 kb/s, 25k tbn, 25k tbc
    Metadata:
      creation_time   : 2013-11-10 15:04:33
    Stream #0.1(eng): Audio: pcm_s16le, 48000 Hz, stereo, 1536 kb/s
    Metadata:
      creation_time   : 2013-11-10 15:04:33
Stream mapping:
  Stream #0:0 -> #0:0 (copy)
  Stream #0:1 -> #0:1 (copy)
Press ctrl-c to stop encoding
frame= 1814 fps=375 q=-1.0 Lsize=  448182kB time=72.96 bitrate=50322.1kbits/s
video:391654kB audio:13688kB global headers:0kB muxing overhead 10.568840%
```

Les arguments utilisés sont les suivants :

- ⇨ **`-i`** indique le fichier à lire, sur lequel effectuer les traitements ;
- ⇨ **`-ss`** précise la partie à « sauter » en début de vidéo ;
- ⇨ **`-t`** précise la durée de la vidéo : on soustrait alors la fin voulue du début pour obtenir la durée cible ;
- ⇨ **`-codec`** permet d'indiquer que l'on souhaite conserver le même codec et ne pas réencoder la vidéo ;
- ⇨ enfin, on donne en dernier le nom du fichier cible.

On peut également noter différentes options qui seraient utilisables :

- ⇨ **`-y`** permet de forcer l'écrasement du fichier cible s'il existe – sans cela, **`avconv`** demande confirmation de l'écrasement, ce qui peut être embêtant lorsque l'on fait de nombreuses tentatives pour un même fichier ;
- ⇨ **`-an`** permet d'indiquer d'ignorer le flux audio ;
- ⇨ **`-codec`** peut prendre comme argument un nom de codec afin de transcoder la vidéo ;
- ⇨ lorsque l'on utilise un autre codec (et qu'on transcode la vidéo), on peut utiliser :
  - **`-b`** pour préciser le *bitrate* d'un flux,
  - **`-s`** pour préciser la taille (en pixels) de la vidéo résultante.

Reprenons la même vidéo. La copie que l'on a faite plus haut est un peu grosse : plus de 400 Mo pour à peine plus d'une minute ! C'est bien pour du traitement local, pour créer une vidéo en HD à graver sur un Blu-ray… mais pour une distribution sur Internet, c'est ingérable. On peut alors utiliser la commande suivante :

Terminal

```
$ avconv -i MVI_4764.MOV -ss 00:00:25 -t 00:01:13 -s 1280x720 -b:v 4M -b:a 64k -codec:v
mpeg4 -codec:a libmp3lame extrait.avi
[...]
    Stream #0.0(eng): Video: mpeg4, yuv420p, 1280x720, q=2-31, 4000 kb/s, 25 tbn, 25 tbc
[...]
    Stream #0.1(eng): Audio: libmp3lame, 48000 Hz, stereo, s16, 64 kb/s
[...]
video:35808kB audio:571kB global headers:0kB muxing overhead 0.342668%
```

Et voilà, le fichier fait maintenant moins de 40 Mo, pour 1 minute et 13 secondes en haute définition.

Ici, on a choisi :

- ⇨ **`-s 1280x720`** : la définition correspondant à la norme « 720p » ;
- ⇨ **`-codec:v mpeg4`** : la vidéo est codée en MPEG4 ;
- ⇨ **`-b:v 4m`** : un bitrate vidéo de 4Mb/s – cela reste une très bonne qualité ;

⇨ `-codec:a libmp3lame` : le son est codé en MP3 ;

⇨ `-b:a 64k` : un bitrate audio de 64kb/s – c'est une qualité audio assez basse, mais la vidéo en question ayant été tournée sur une plage, le son reste assez médiocre.

Bien sûr, pour avoir un fichier plus petit, on aurait pu réduire le bitrate vidéo.

Par exemple, avec un bitrate vidéo de 500kb/s, on obtient un fichier de moins de 6 Mo... mais au détriment de la qualité de l'image. Cette baisse de qualité peut être compensée par la réduction de la définition (640x360 par exemple), mais on n'est alors plus en haute définition ; il y a une limite au gain de place...

Vous pouvez obtenir la liste complète des codecs disponibles avec la commande suivante :

Terminal
```
$ avconv -codecs
```

Notez que nous n'allons pas effectuer la suite des opérations sur le fichier `extrait.avi` : afin de minimiser les pertes, il est préférable de travailler exclusivement à partir de la vidéo d'origine et non de travailler par étapes. C'est d'ailleurs comme cela que fonctionnent les logiciels graphiques : on définit successivement différentes opérations à effectuer sur les vidéos, mais c'est en une fois que le logiciel les applique, quand on lui demande de générer le résultat...

2. INSÉRER UN FILIGRANE

Imaginons que l'on veuille intégrer un filigrane à la vidéo, afin de montrer qu'on en est bel et bien l'auteur.

Bien sûr, un filigrane dans une vidéo c'est parfois gênant (qui n'a jamais pesté contre le logo d'une chaîne TV qui cache des bouts de l'image ?), mais il y a des cas où cela se justifie pleinement : pensons par exemple au monteur professionnel qui veut présenter son travail sur son site web, à destination d'éventuels recruteurs : cela permet d'éviter qu'une autre personne ne s'attribue son travail, ça peut contribuer à protéger sa vitrine virtuelle.

On dispose d'un fichier `logo.png`, à fond transparent, qui fait 150×150 pixels. L'inclure en surimpression nécessite une option relativement simple :

Terminal
```
-vf 'movie=logo.png [watermark];[in][watermark]overlay=10:10[out]'
```

`-vf` est l'option qui demande d'appliquer un filtre vidéo (**v**ideo **f**ilter) ; son argument contient le ou les filtre(s) vidéo à appliquer. Décortiquons cette option :

⇨ `movie=logo.png [watermark]` définit un nouveau flux vidéo, utilisant le fichier `logo.png` et nommé `watermark` ;

⇨ `[in][watermark]overlay=10:10[out]` indique qu'il faut injecter la vidéo nommée `watermark` et appliquer le filtre `overlay`, lui donnant comme argument `10:10` : il positionne alors la vidéo en incrustation à 10 pixels de la gauche et 10 pixels du haut de la vidéo.

À retenir

Que ce soit pour un filigrane ou pour d'autres transformations, il est obligatoire de choisir un codec de sortie et non l'option `copy`. En effet, lorsque l'on choisit `copy`, `avconv` utilise le flux vidéo entrant sans le modifier, les traitements vidéo ne se font donc simplement pas.

On parle bien de vidéo à incruster et non d'image : **avconv** travaille sur des flux vidéo et traite une image comme telle : une vidéo qui présente constamment la même image.

Les noms de flux ou de points de jointure, indiqués entre crochets (**[** et **]**), sont libres : ils permettent de définir comment s'articulent les filtres entre eux. Seuls deux noms sont réservés : **[in]** définit le début de la chaîne, là où la vidéo source entre et **[out]** définit la sortie de la chaîne, là où est pris le résultat à stocker dans le fichier cible.

Notre commande devient donc :

Terminal
```
$ avconv -i MVI_4764.MOV -ss 00:00:25 -t 00:01:13 -s 1280x720 -b:v
4M -b:a 64k -codec:v mpeg4 -codec:a libmp3lame -vf 'movie=logo.png
[watermark];[in][watermark]overlay=10:10[out]' filigrane.avi
[...]
video:35804kB audio:571kB global headers:0kB muxing overhead 0.342671%
```

3. DES FILTRES UN PEU PLUS COMPLEXES…

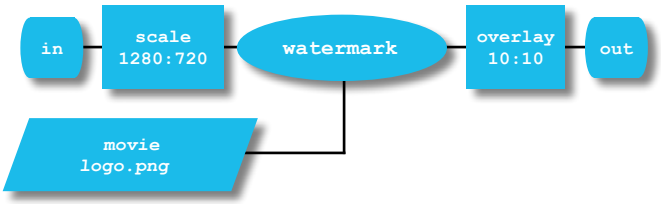
Faisons une petite digression sur les filtres vidéo : il est nécessaire de bien les comprendre pour percevoir la puissance de Libav. Un filtre vidéo est donc défini par un ensemble de chaînes, dont les définitions sont séparées par des doubles-points (**;**). Dans chaque chaîne, on peut définir des points de jointure, pour effectuer des traitements en parallèle. Par ailleurs, dans chaque chaîne, il faut séparer les différents filtres par des virgules (**,**).

Mais avant tout, penchons-nous sur un outil très proche de **avconv**… Faire des expérimentations vidéo sur des fichiers existants est assez pénible : on définit les filtres, on exécute la commande, on attend que la vidéo soit transformée, on la visionne, on corrige nos effets… Et si on pouvait voir directement nos modifications ? C'est ce que propose la commande **avplay** : on lui donne en argument le nom du fichier à lire et les transformations à y effectuer. Par contre, certaines options seront alors refusées, c'est le cas de **-codec** (c'est évident : on lit la vidéo à l'écran, on ne l'écrit pas dans un fichier, il n'y a pas à choisir de codec de sortie !) mais encore de **-s**, celle-ci étant traitée lors du transcodage de la vidéo : pas de transcodage, pas de modification de la taille ! Pour la modification de la taille de l'image, on peut très bien utiliser le filtre vidéo **scale**, avec **scale=1280:720**.

Avec la commande suivante, on obtient alors le même résultat (vidéo en 1280×720 et incrustation d'un filigrane), directement à l'écran, sans transcodage et sans écriture dans un fichier :

Terminal
```
$ avplay -vf 'movie=logo.png [watermark];[in] scale=1280:720, [watermark]
overlay=10:10 [out]' MVI_4764.MOV
```

Cette définition de filtre peut être décrite par le schéma suivant :

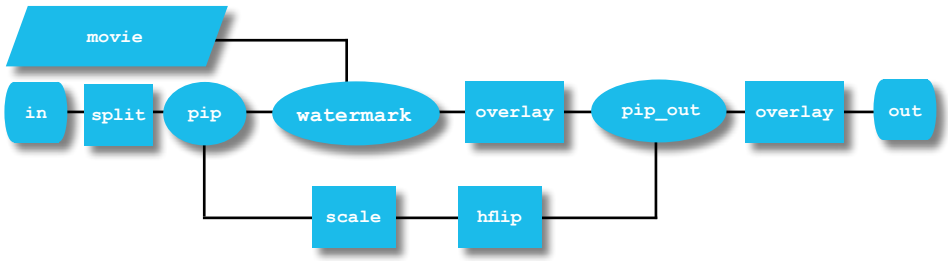


Pour nous amuser, nous pouvons nous baser sur autre chose qu'un fichier… Vous avez une webcam ? Alors utilisons son périphérique, qui est probablement **/dev/video0** – il faut forcer l'utilisation de **video4linux2**, avec l'option **-f** :

Terminal

```
$ avplay -vf 'movie=logo.png [watermark];[in] [watermark] overlay=10:10
[out]' -f video4linux2 /dev/video0
```

Maintenant, imaginons que l'on souhaite reproduire la même image, la réduire et la permuter horizontalement (effet « miroir »), la placer dans le coin en bas à droite (un peu comme la fonctionnalité « PIP » (*picture in picture*) de certaines télévisions), tout en ne conservant le filigrane que sur l'image principale. L'articulation de l'ensemble est alors décrite par le schéma suivant :



L'ensemble de filtres correspondant à ce schéma est :

- `movie=logo.png [watermark]` ;
- `[in] split [pip], [watermark] overlay=10:10, [pip_out] overlay=W-w-10:H-h-10 [out]` ;
- `[pip] scale=128:96, hflip [pip_out]`.

La commande est alors :

Terminal

```
$ avplay -vf 'movie=logo.png [watermark]; [in] split [pip], [watermark]
overlay=10:10, [pip_out] overlay=W-w-10:H-h-10 [out]; [pip] scale=128:96,
hflip [pip_out]' -f video4linux2 /dev/video0
```

Il reste encore un point qui n'est pas très clair dans ces filtres : l'argument au deuxième `overlay`. Pour celui-ci, l'explication est simple :

- `W` = largeur de l'image principale ;
- `w` = largeur de l'image à incruster ;
- `H` = hauteur de l'image principale ;
- `h` = hauteur de l'image à incruster.

Avec `W-w-10`, on place le coin « haut gauche » de l'image incrustée à la largeur de l'image principale, moins la largeur de l'image incrustée moins 10 pixels. Le raisonnement est le même pour la hauteur. Le résultat est alors que le coin « bas droit » de l'image incrustée est à 10 pixels du coin « bas droit » de l'image principale...

Pour obtenir la liste des filtres disponibles, on peut utiliser la commande suivante ;

Terminal

```
$ avplay -filters
```

Petit exercice : réussirez-vous à dessiner le schéma correspondant à la commande suivante ?

Terminal

```
$ avplay -vf '[in] split [T1], split [T3], fade=in:5:60, [T4] overlay=W-
w-10:H-h-10, [T2] overlay=10:10 [out]; [T1] scale=256:192, hflip,
lutrgb=g=0:b=0 [T2]; [T3] scale=128:96, negate, vflip [T4]' -f video4linux2 /
dev/video0
```

Réponse en fin d'article...

À savoir

Avec le projet FFmpeg, on a également à notre disposition le filtre `mp`, qui permet de faire appel à des transformations effectuées par MPlayer ; en effet, les développeurs de FFmpeg sont très proches de ceux de MPlayer (certains travaillent sur les deux projets). Les développeurs de Libav, de leur côté, ne veulent pas dépendre d'un autre logiciel, ce filtre n'y est donc pas disponible…

4. TITRER LA VIDÉO

On souhaite maintenant ajouter un titre au début de la vidéo. Sans aller jusqu'à une animation à la *Star Wars*, plaçons simplement un texte statique en début de vidéo, avec un petit fondu d'introduction.

Tout d'abord, il faut créer l'image qui contiendra ce texte. En effet, Libav ne sait pas travailler avec du texte. Par contre, on connaît un logiciel qui sait faire des images et qui sait travailler avec du texte : ImageMagick !

Notre exemple sera une image de 1920×1080 pixels, présentant un fond dégradé radial du gris clair vers le noir. Le texte, centré et de couleur bleu/gris clair, aura une ombre portée donnant à l'ensemble un petit effet de relief. Pour cela, nous utiliserons les commandes `convert` (pour générer le texte souhaité) et `composite` (pour y inclure un dégradé). Tout cela en une seule ligne :

Terminal

```
$ convert -size 1920x1080 xc:none -gravity Center -pointsize 90 -stroke
black -fill black -strokewidth 5 -draw "text 5,5 'Cerf-volants à la
plage'" -channel RGBA -blur 12x4 -stroke none -fill LightSteelBlue2 -draw
"text 0,0 'Cerf-volants à la plage'" png:- | composite -size 1920x1080
radial-gradient:grey62-black -compose dst-over - png24:titre.png
```

Le résultat est le suivant :

Décortiquons les arguments de ces commandes :

- `-size 1920x1080` (utilisé pour chaque commande) : on définit la taille de l'image sur laquelle travailler ;
- `xc:none` : on crée un canevas vide ;
- `-gravity Center` : tout élément plus petit que l'image y sera centré ;
- `-pointsize 90` : le texte fera 90 pixels de haut ;
- `-stroke black` : on dessine avec un trait noir ;
- `-fill black` : on dessine avec un remplissage noir ;
- `-strokewidth 5` : le trait a une largeur de 5 pixels ;
- `-draw «text 5,5 '[...]'»` : on dessine le texte en question, décalé de 5 pixels vers la droite et 5 vers le bas ;
- `-channel RGBA` : on travaille sur quatre canaux (incluant le canal alpha pour la transparence) au lieu des trois habituels ;
- `-blur 12x4` : on donne un effet de flou sur un rayon de 4 pixels, en travaillant sur des échantillons de 12 pixels ;

- ⇨ **-stroke none** : pour le prochain dessin, on ne dessine pas le trait ;
- ⇨ **-fill LightSteelBlue2** : le remplissage du prochain dessin sera bleu/gris clair ;
- ⇨ **-draw «text 0,0 '[...]'»** : on dessine le texte en question centré sur la page ;
- ⇨ **png:-** : on écrit cette image au format PNG sur la sortie standard ;
- ⇨ **radial-gradient:grey62-black** : sur une nouvelle image, on crée un dégradé radial du gris clair au noir ;
- ⇨ **-compose dst-over -** : on superpose l'image provenant de l'entrée standard (**-**) à l'image générée ;
- ⇨ **png24:titre.png** : on écrit cette image au format « PNG 24 bits » dans le fichier **titre.png**.

Le format PNG 24 bits attribue 8 bits à chaque canal : en effet, Libav ne travaille qu'avec 8 bits par canal, alors que le PNG attribue par défaut 16 bits par canal.

Une fois cette image créée, il faut la transformer en vidéo. Pour cela, on utilisera encore une fois la commande **avconv** :

Terminal

```
$ avconv -loop 1 -i titre.png -t 00:00:10 -vf 'fade=in:25:125' -b 4M -codec mpeg4
titre.avi
[...]
Input #0, image2, from 'titre.png':
  Duration: 00:00:00.04, start: 0.000000, bitrate: N/A
    Stream #0.0: Video: png, rgb24, 1920x1080, 25 fps, 25 tbr, 25 tbn, 25 tbc
[...]
Output #0, avi, to 'titre.avi':
[...]
    Stream #0.0: Video: mpeg4, yuv420p, 1920x1080, q=2-31, 4000 kb/s, 25 tbn, 25 tbc
[...]
```

On utilise les arguments suivants :

- ⇨ **-loop 1** : lit le fichier en boucle à l'infini ;
- ⇨ **-i titre.png** : on traite le fichier **titre.png** en entrée ;
- ⇨ **-t 10** : l'écriture s'arrête au bout de 10 secondes ;
- ⇨ **-vf 'fade=in:25:125'** : on crée un fondu d'entrée, qui commence au bout de 25 images (1 seconde) et qui dure 125 images (5 secondes) ;
- ⇨ **-b 4M** : la vidéo résultante aura un bitrate de 4 Mo/s (à l'identique de l'autre vidéo, histoire d'avoir une qualité équivalente) ;
- ⇨ **-codec mpeg4** : on utilise le même codec que pour l'autre vidéo, pour la même raison que ci-dessus.

Pour l'étape suivante, la commande adaptée serait :

Terminal

```
$ avconv -i 'concat:titre.avi|filigrane.avi' -codec copy video.avi
```

… mais celle-ci pose un problème, car si la seconde vidéo a bien des pistes audio, la première n'en a pas ; du coup, lors de la création du fichier **video.avi**, aucune piste audio n'est incluse. En inversant les noms des fichiers, on obtient bien le résultat prévu, son y compris… mais notre objectif est de mettre le titre au début, pas à la fin !

La solution consiste alors à générer 10 secondes de silence avec la commande **sox** (du paquet **sox**, ou n'importe quel autre outil capable de générer du silence), puis de

créer le fichier **titre.avi** en incluant ce silence et enfin, de concaténer les deux fichiers, incluant cette fois-ci le son :

Terminal

```
$ sox -n -r 48000 -c 2 silence.wav trim 0 10
$ avconv -loop 1 -i titre.png -i silence.wav -t 00:00:10 -vf 'fade=in:25:125' -b:v
4M -b:a 64k -codec:v mpeg4 -codec:a libmp3lame titre.avi
[...]
Output #0, avi, to 'titre.avi':
[...]
    Stream #0.0: Video: mpeg4, yuv420p, 1920x1080, q=2-31, 4000 kb/s, 25 tbn, 25 tbc
    Stream #0.1: Audio: libmp3lame, 48000 Hz, stereo, s16, 64 kb/s
[...]
$ avconv -i 'concat:titre.avi|filigrane.avi' -codec copy video.avi
[...]
```

Et voilà, nous avons notre vidéo qui répond aux contraintes suivantes :

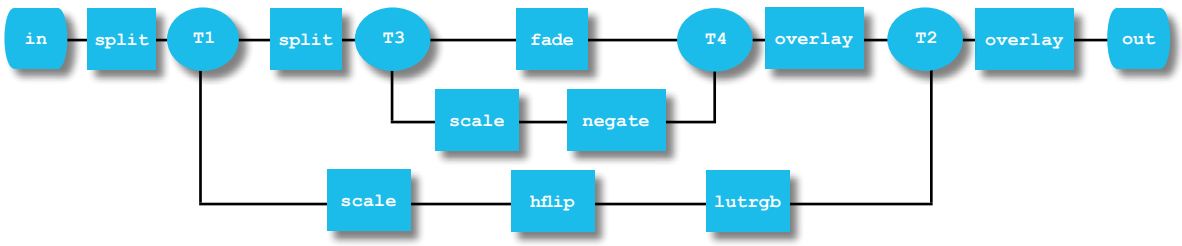
- un fondu d'entrée ;
- un titre textuel fixe pendant quelques secondes ;
- une vidéo avec un filigrane.

5. ALLER PLUS LOIN

Comme vous l'avez probablement compris avec les articles précédents, encore une fois on n'a fait qu'effleurer les possibilités qu'offrent Libav ou FFmpeg : à vous de lire les pages de manuel, vous documenter et expérimenter afin de terminer peut-être par un montage vidéo complet, uniquement en ligne de commandes et automatisé. Vous voulez une idée ? Si vous avez une webcam, un panorama intéressant et un site web, vous pourrez proposer de manière automatique à vos visiteurs le lever de soleil sur votre superbe panorama tous les jours, avec archivage à la clé !

6. RÉPONSE À L'EXERCICE

Alors, vous avez réussi à dessiner le schéma de l'exercice donné plus haut ?
Voici le corrigé :



On obtient :

- en grand, l'image de la webcam avec un fondu de 60 images après 5 images noires ;
- en incrustation en haut à gauche, la même image réduite à 256×192 et à laquelle on a supprimé les canaux vert et bleu ;
- en incrustation en bas à droite, la même image réduite à 128×96 en négatif.

Le tout bien entendu en direct, l'image prise de la webcam est affichée en temps réel à l'écran ! Vous imaginerez aisément tout ce que l'on peut faire avec Libav ou avec FFmpeg ! ■

LINUX MAGAZINE / FRANCE
LINUX PRATIQUE
MISC Multi-System & Internet Security Cookbook
Open Silicium

VENEZ DÉCOUVRIR NOS GUIDES !

DÉJÀ PARUS !

DISPONIBLES CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

boutique.ed-diamond.com

Les classiques

Les outils indispensables pour manipuler et sécuriser ses données

Matériel

Des commandes pour identifier précisément et exploiter son matériel

Système

Diagnostiquer les problèmes, virtualiser ses systèmes et opter pour la flexibilité

Net & Sécurité

Mise en place d'une connexion SSH, contrôle à distance et chiffrement de flux

Multimédia

Retour sur deux incontournables du multimédia : FFmpeg et la suite ImageMagick