ROYAUME DU MAROC

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Commandes Gnu/Linux de base

www.ofppt.info

OFPPT

DIRECTION **R**ECHERCHE ET **I**NGENIERIE DE **F**ORMATION

SECTEUR NTIC

Sommaire

1. Login et déconnexion

Comme il l'a déjà été dit, Linux est un système multi-utilisateur. Il faut donc que chacun s'identifie pour que le système puisse fonctionner correctement. Cela est réalisée lors de l'opération dite de *login* (du verbe anglais « to log in », qui signifie « s'enregistrer » dans le système). Le login consiste essentiellement à taper son nom d'utilisateur, valider, et répondre éventuellement à la demande de mot de passe de la part du système.

Le login doit être la première opération à effectuer. Il est impossible d'accéder au système d'une autre manière, et la vérification du mot de passe fournit l'authenticité de l'utilisateur qui se logue. Ainsi, le système sait en permanence au nom de quelle personne il effectue les opérations demandées. Cette opération est à la base des mécanismes de sécurité et de personnalisation du système pour chaque utilisateur.

Il existe deux types de login. Le plus courant est le login en mode texte, qui peut être fait directement sur le poste local ou à travers un réseau. Le système vous invite à vous identifier avec la ligne suivante :

login:

D'autres informations peuvent être affichées avant le mot login, qui peuvent vous renseigner sur la nature du système. Quoi qu'il en soit, vous devez taper votre nom d'utilisateur (que l'on appelle simplement « login »), ou « root » si vous désirez vous connecter en tant qu'administrateur. Le système vous demande alors le mot de passe avec la ligne suivante :

password:

Bien entendu, vous ne devrez jamais oublier votre mot de passe administrateur. Si toutefois cela vous arrive, vous n'aurez plus qu'une seule solution : démarrer l'ordinateur à partir d'une disquette système, et faire le ménage dans le fichier de mots de passe. Cette opération n'est jamais très agréable à réaliser. Conclusion : n'oubliez jamais votre mot de passe.

Le deuxième type de login est le login graphique, sous X11. Ce type de login a en général lieu sur un terminal X (c'est-à-dire un terminal graphique). La procédure peut varier selon l'environnement utilisé, mais le principe reste toujours le même : il faut fournir son nom d'utilisateur et son mot de passe.

Si, comme la plupart des gens, vous ne cherchez pas à utiliser votre ordinateur à distance à travers un réseau, vous vous connecterez quasiment toujours en local. Linux fournit, pour l'utilisateur local, plusieurs terminaux virtuels. Cela signifie qu'il est possible de se connecter plusieurs fois dans le système dans des terminaux différents. Pour passer d'un terminal virtuel à un autre, il suffit de taper les combinaisons de touches ALT+DROITE ou ALT+GAUCHE, où DROITE et GAUCHE sont respectivement les flèches du curseur droite et gauche. Il est également possible d'accéder à un terminal donné à l'aide de la combinaison de

touches ALT+Fn, où Fn est l'une des touches de fonction F1, F2, F3, etc. La plupart des distributions utilisent au moins quatre terminaux virtuels, plus un terminal X. Le terminal X est le terminal graphique, qui fonctionne sous XWindow. Vous noterez sans doute que lorsqu'on est sous XWindow, les combinaisons ALT+Fn ont une autre signification. Elles ne peuvent donc pas être utilisées pour basculer vers les terminaux en mode texte. Pour remédier à ce problème, une autre combinaison de touches a été définie, spécialement pour XWindow : CTRL+ALT+Fn. Il suffit donc simplement d'utiliser la touche CTRL en plus de la touche ALT.

L'utilisation des terminaux virtuels est très pratique, même pour un seul utilisateur. En effet, ceux-ci permettent de lancer plusieurs programmes simplement, à raison d'un par terminal virtuel, et de s'y retrouver ainsi plus facilement. Pour ceux qui ne connaissent pas les systèmes Unix, il est recommandé de jouer un peu avec les terminaux virtuels afin de simuler la présence de plusieurs utilisateurs. Ils auront ainsi un aperçu de la puissance de ces systèmes.

Lorsqu'on a fini de travailler, il faut se déconnecter. Cette opération est très simple pour les terminaux non graphiques, puisqu'il suffit de taper la commande suivante :

logout

Si d'aventure cette commande ne fonctionnait pas, vous pourrez utiliser la commande **exit** ou la combinaison de touches CTRL+d, qui terminent le shell courant (y compris le shell de login).

Pour les terminaux X, le processus de déconnexion dépend de l'environnement utilisé. Il faut tâtonner un peu, et normalement on trouve une option de menu du style « logout » ou « déconnexion ». Vous pouvez par exemple cliquer sur le bouton droit de la souris sur le bureau de l'environnement de travail, afin d'appeler un menu contextuel. Dans bien des cas, ce menu contient une option de déconnexion.

Il est très important de se déconnecter et de ne jamais laisser une session ouverte. En effet, cette négligence peut vous coûter cher, car une personne mal intentionnée pourrait très bien utiliser ce terminal à vos dépends. Il aurait tous vos droits, et effectuerait ses opérations en votre nom. La sécurité du système garantissant que vous seul pouvez vous connecter sous ce nom, grâce au mot de passe, vous seriez donc responsable des agissements de l'intrus. Bien entendu, ce genre de considération n'a pas autant d'importance pour un particulier que dans une entreprise ou une collectivité quelconque.

2. Arrêt et redémarrage du système

Il faut bien comprendre que Linux, tout comme la plupart des systèmes d'exploitation modernes, ne peut pas être arrêté en éteignant directement l'ordinateur, comme on le faisait autrefois avec le DOS. En effet, la plupart des systèmes d'exploitation utilisent une partie de la mémoire de l'ordinateur pour y stocker temporairement les données qui ont été lues à partir du disque et celles qui doivent y être écrites. Cette zone de mémoire constitue ce qu'on appelle un *tampon* (« buffer » en anglais), et elle sert à accélérer les accès aux périphériques plus lents, que sont les disques durs et lecteurs de CD-ROM. Il va de soi qu'une requête de lecture sur des données déjà situées en mémoire est infiniment plus rapide que si elles ne s'y trouvaient pas. Il est en revanche plus difficile de comprendre pourquoi les requêtes d'écriture doivent être différées. La raison est la suivante : le système préfère différer l'écriture physique sur le disque parce qu'une autre requête d'écriture dans la même zone du disque peut très bien avoir lieu ultérieurement. Si les données qui n'ont pas été écrites sont ainsi modifiées par une requête ultérieure, il n'est plus nécessaire de les écrire, et ainsi le système peut économiser un temps précieux en ne le faisant pas. Si les données à écrire sont contiguës à celles d'une requête précédente, le système peut les écrire en bloc, ce qui est toujours plus rapide que de faire plusieurs écritures partielles (notamment parce que les têtes de lecture du disque n'ont pas à être déplacées). Enfin, si les données qui doivent être écrites font l'objet d'une requête de lecture, il va de soi qu'elles sont immédiatement accessibles. On voit que cette stratégie permet de travailler beaucoup plus vite. De facto, Linux utilise toute la mémoire vive libre pour ses tampons d'entrées / sorties, ce qui en fait un système extrêmement performant. Le gain en performances peut facilement atteindre un facteur 3 ou 4.

Le problème majeur est évidemment que si l'on éteint l'ordinateur brutalement, les données dont l'écriture a été différée sont perdues. Pire, parmi ces données, il est probable qu'il y ait des informations vitales pour le système de fichiers, ce qui fait qu'il risque fort d'être endommagé. Les systèmes de fichiers journalisés comme EXT3 et ReiserFS sont à l'abri de ce type d'erreur en raison de l'accès transactionnel aux structures de données du systèmes de fichiers qu'ils utilisent, et le système parvient généralement à réparer les autres systèmes de fichiers lors de la vérification qui est lancée au redémarrage suivant de la machine, mais il est inutile de prendre des risques. Tout cela signifie qu'il est impératif de prévenir le système avant de l'arrêter, pour qu'il puisse écrire les données situées dans ses tampons.

L'arrêt du système est une opération qui est du ressort de l'administrateur. On ne peut donc le réaliser que sous le compte root. Plusieurs commandes sont disponibles, les plus simples sont données ci-dessous :

- **halt**, qui permet d'arrêter le système.
- **reboot**, qui permet de le redémarrer.

Ces commandes sont en fait des scripts permettant d'effectuer les opérations d'arrêt et de redémarrage du système rapidement. Si elles ne sont pas disponibles sur votre distribution, vous devrez sans doute utiliser la commande

générique suivante :

shutdown [-r] now
: où l'option -r permet de demander un redémarrage et non un arrêt simple.

Il est également possible que votre gestionnaire de bureau vous donne le moyen d'arrêter l'ordinateur par l'intermédiaire de l'interface graphique de X11. La technique à utiliser dépend évidemment de l'environnement que vous aurez installé, et elle ne sera pas décrite ici. Consultez la documentation de votre distribution pour plus de détails à ce sujet. De plus, la plupart des distributions provoquent un redémarrage de la machine lorsqu'on appuie sur les touches CTRL+ALT+SUPPR simultanément dans un terminal virtuel.

3. Pages de manuel

Maintenant que vous savez l'essentiel pour conserver votre système en bon état, nous allons traiter des autres commandes Unix. Parmi elles, il en est qui sont certainement fondamentales : ce sont les commandes qui permettent d'obtenir de l'aide !

Chaque commande Unix a une page de manuel qui la décrit. Ces pages sont très souvent écrites en anglais, mais elles sont très précises et fournissent toutes les informations dont on peut avoir besoin. Pour afficher la page de manuel d'une commande, il suffit d'utiliser la commande suivante :

man page
: où page est la page de manuel de la commande sur laquelle on cherche des informations. En général, le nom de la page de manuel est le même que celui de la commande. Par exemple, pour afficher l'aide sur la commande **cp**, il suffit de taper :

man cp

Lorsqu'une page de man est affichée, il est possible de faire défiler son texte à l'aide des touches du curseur. Pour quitter l'aide, il suffit d'appuyer sur la touche q.

Les pages de man sont classées en groupes de pages thématiques, chaque groupe étant identifié généralement par un numéro ou une lettre. Si la page de man affichée ne correspond pas à celle désirée, c'est qu'une page homonyme d'un autre groupe a été utilisée. Dans ce cas, il faut préciser l'identificateur du groupe de pages de manuel avant le nom de la page à afficher :

man groupe page
: où groupe est l'identificateur du groupe auquel la page de manuel appartient. Les principaux groupes sont les suivants :

Tableau 3-1. Groupes de pages de man

Identificateur	Type de pages de manuel
1	Commandes utilisateur
2	Appels systèmes (programmation en C)
3	Fonctions de la bibliothèque C
4	Description des fichiers spéciaux
5	Description des fichiers de configuration
6	Jeux et programmes divers
7	Programmes systèmes divers
8	Administration système

Si vous ne savez pas dans quel groupe se trouve une page de manuel, vous pouvez utiliser l'option -k, qui permet d'afficher l'ensemble des pages disponibles portant ce nom :

man -k commande

L'identificateur du groupe est en général précisé entre parenthèses, à la suite du nom de la page de manuel.

Il se peut également que vous recherchiez de l'aide sur un sujet donné, mais que vous ne connaissiez pas le nom exact de la page de manuel qui en parle. Pour ce genre de recherche, vous pourrez utiliser le programme **apropos**, qui recherchera toutes les pages de manuel qui contiennent un mot clé particulier. Ce programme s'utilise avec la syntaxe suivante :

apropos mot
où mot est le mot clé à rechercher dans toutes les pages de manuel.

La commande **man** est la commande d'aide standard sur tous les systèmes Unix. Cependant, Linux utilise un grand nombre de commandes écrites sous la licence GNU, et qui utilisent un format d'aide spécifique à GNU. L'aide pour ces commandes peut être obtenue par la commande suivante :

info commande

Il se peut que les deux méthodes fonctionnent. Dans ce cas, la page de man sera certainement moins récente que la page d'info, car la commande que vous utilisez est sans aucun doute une commande GNU, qui a été fournie avec sa page d'information. Il est donc recommandé de lire plutôt la page d'information GNU.

Le format d'aide GNU est plus riche que celui de man, puisqu'il permet de naviguer dans le système d'aide à l'aide de liens hypertextes. Ces liens sont organisés hiérarchiquement, avec des chapitres et des sous-chapitres. Chaque chapitre dispose d'une forme de table des matières constituée de menus, qui

permettent d'accéder aux sous-chapitres. Les menus se distinguent du texte normal par un astérisque (« * ») en début de ligne dans la table des matières. Les commandes clavier suivantes pourront vous être utiles pour naviguer dans la hiérarchie du système d'aide de GNU :

- la touche de tabulation permet de passer au lien hypertexte suivant ;
- la touche n (pour « Next ») permet de passer au chapitre suivant ;
- la touche p (pour « Previous ») permet de revenir au chapitre précédent ;
- la touche u (pour « Up ») permet de remonter d'un niveau dans le système d'aide et d'atteindre la table des matières référençant le chapitre courant.

Enfin, la commande q permet de quitter le système d'aide.

4. Opérations de base sur les répertoires

Ce paragraphe va vous décrire les opérations de base qu'il faut savoir faire pour manipuler les répertoires du système de fichiers.

La première commande est évidemment celle qui permet de lister le contenu d'un répertoire. Elle dispose d'un grand nombre d'options :

ls [options] [fichier]

où fichier est le nom d'un fichier ou d'un répertoire que l'on désire lister. Si ce paramètre est absent, **ls** affichera tous les fichiers du répertoire courant. Les principales options sont -l, qui permet d'afficher des informations étendues (notamment les propriétaires, les groupes, les droits, la taille et éventuellement les liens), et -a, qui permet d'afficher tous les fichiers, y compris les fichiers cachés (ceux dont le nom commence par un point).

La deuxième commande est celle qui permet de changer de répertoire courant. Sa syntaxe est très simple :

cd [chemin]

où chemin est un chemin de répertoire Unix valide. Ce chemin est constitué des noms des répertoires et sous-répertoires successifs, séparés par des barres obliques « / ». Si aucun chemin n'est spécifié, cette commande change le répertoire courant pour le répertoire personnel de l'utilisateur. Par exemple, pour aller dans le répertoire d'installation de XWindow, il faut taper la commande suivante :

cd /usr/X11

La notion de chemin sera détaillée dans le paragraphe suivant. « cd » est l'abréviation de l'anglais « Change Directory ».

La troisième commande permet de créer un répertoire :

mkdir chemin

où chemin est le chemin spécifiant le répertoire à créer. Si le chemin ne

contient que le nom du répertoire à créer, celui-ci est créé dans le répertoire courant et devient donc un sous-répertoire. « mkdir » est l'abréviation de l'anglais « MaKe DIRectory »).

La commande pour supprimer un répertoire est la suivante :

rmdir chemin

Pour supprimer un répertoire, il faut qu'il soit vide (c'est-à-dire qu'il ne contienne ni fichier, ni répertoire). « rmdir » est l'abréviation de l'anglais « ReMove DIRectory ».

Enfin, voici une commande dont vous ne vous servirez normalement que très peu, voire pas du tout. Elle permet d'afficher le répertoire courant :

pwd

Cette commande n'est a priori pas très utile, car le shell affiche toujours le répertoire courant sur la plupart des distributions. Cependant, le chemin affiché par le shell étant relatif au répertoire personnel de l'utilisateur lorsqu'on se trouve dans un sous-répertoire de celui-ci, la commande **pwd** peut être utile lorsqu'on désire obtenir un chemin absolu sur le répertoire courant. « pwd » est l'abréviation de l'anglais « Print Working Directory ». Cette commande est également utilisée par les scripts pour déterminer le répertoire à partir duquel ils sont exécutés.

5. Notions sur les chemins Unix

Les chemins Unix permettent de qualifier complètement un répertoire ou un fichier sur le disque. Comme on l'a vu, ils utilisent pour cela les noms de ces répertoires et de ces fichiers, et ils les combinent pour indiquer comment atteindre la cible dans le système de fichiers. Sous Unix, le séparateur utilisé pour séparer les différents constituants du nom est la barre oblique (« / »). Le répertoire racine n'a pas de nom, et peut donc être référencé par une barre oblique seule.

Les chemins peuvent être *absolus* (c'est-à-dire qu'ils peuvent partir du répertoire racine) ou *relatifs* (c'est-à-dire qu'ils peuvent partir du répertoire courant). Si l'on utilise un chemin relatif, il faut savoir que le répertoire courant est désigné par un point (« . »), et que le répertoire parent du répertoire courant est désigné par deux points successifs (« .. »). Ainsi, si l'on est dans le répertoire /usr/local/bin, on peut accéder au répertoire /usr/X11/bin avec les deux chemins suivants :

```
/usr/X11/bin
        ou :
../../X11/bin
```

Le premier chemin est absolu, parce qu'il part directement du répertoire racine. Le deuxième chemin est relatif, car il part du répertoire courant.

Note : Il va de soi que les chemins relatifs ne sont valides, sauf coup de chance, que dans le répertoire dans lequel ils sont écrits, alors que les chemins absolus sont toujours valables. En revanche, si des répertoires sont déplacés ensemble, les chemins relatifs entre ces répertoires restent valides, mais les chemins absolus deviennent faux. Toutefois, ces considérations ne concernent pas un utilisateur de base.

La plupart des shells sont capables d'effectuer ce que l'on appelle la *complétion automatique* des commandes. La complétion automatique permet de n'écrire qu'une partie des noms de fichiers ou de répertoires et de demander au shell de compléter ces noms. Cela peut se faire de deux manières. La première solution, qui est aussi la plus simple, consiste à taper le début du nom, puis d'utiliser une touche spéciale qui permet de demander au shell de le compléter. Si vous utilisez le shell bash (bash est le shell de prédilection sur les systèmes Linux), cette touche est la touche de tabulation. Ainsi, si vous tapez :

cd /ho
: et que vous appuyez sur la touche de tabulation, bash complétera cette ligne de la manière suivante :

cd /home/

Pour cela, il regarde la liste des fichiers et des répertoires qui commencent par « ho » dans le répertoire racine. Normalement, il ne s'y trouve que le répertoire /home/, et c'est ce nom que bash utilise. Il va de soi qu'il ne faut pas qu'il y ait ambiguïté sur un nom partiel. Par exemple, si vous tapez la commande suivante :

cd /usr/l
: et que vous demandiez au shell de compléter le nom, il ne pourra pas choisir quel répertoire utiliser entre /usr/lib/ et /usr/local/. Dans ce cas, il émettra un petit bip signalant l'erreur. En appuyant une fois de plus sur la touche de tabulation, bash affiche la liste des choix possibles et vous propose de terminer la ligne de commande en saisissant des caractères supplémentaires afin de résoudre l'ambiguïté.

La deuxième solution est d'utiliser les caractères génériques du shell. Ces caractères permettent de désigner n'importe quel caractère, ou n'importe quelle séquence de caractères. Ils sont désignés respectivement par un point d'interrogation (« ? ») et par un astérisque (« * »). Ainsi, si l'on tape la commande suivante :

cd /ho*
: le shell ira directement dans le répertoire /home/, car le caractère générique « * » peut être remplacé par la séquence de caractères « me ». Il est également possible d'écrire :

cd /?ome
: et dans ce cas le caractère générique « ? » sera remplacé par « h ». Encore une fois, il ne faut pas qu'il y ait ambiguïté. Dans le cas contraire, le comportement varie selon le shell. En général, il essaie de résoudre l'ambiguïté au mieux en analysant la suite du chemin, et s'il ne peut pas, il affiche un message d'erreur.

Note : Ces caractères génériques sont interprétés directement par le shell et non par la commande qui les reçoit en paramètres. Tout nom de fichier contenant un caractère générique est remplacé par la liste des fichiers qui correspondent au motif donné. S'il n'existe qu'un seul fichier dans cette liste, il est possible d'utiliser les commandes comme cd, qui ne prennent qu'un seul paramètre. Mais il est possible d'utiliser les commandes acceptant plusieurs paramètres, même s'il y a plusieurs fichiers dans cette liste. Ainsi, la commande suivante :

ls *txt

permet de lister tous les fichiers dont le nom se termine par « txt ». Il ne peut évidement pas y avoir d'ambiguïté dans ce cas.

Si on doit passer un paramètre comprenant l'un des caractères génériques interprétés par le shell à une commande particulière, on devra préfixer les caractères génériques d'un caractère d'échappement pour signaler au shell qu'il ne doit pas l'interpréter. Ce caractère d'échappement est la barre oblique inverse (« \ »). Il est également possible de passer les paramètres entre guillemets « " », car le shell n'interprète pas les caractères génériques dans les chaînes de caractères. Par exemple, pour créer un répertoire ?, on utilisera la commande suivante :

mkdir \?

Les noms de fichiers commençant par un tiret (caractère '-') posent également des problèmes avec la plupart des commandes, car ce caractère est utilisé pour spécifier des options. Ce n'est pas le shell qui interprète ce caractère dans ce cas, mais le problème est le même. La plupart des commandes utilisent l'option -- (elle-même introduite par un tiret, ce qui fait donc deux tirets accolés) pour signaler que ce qui suit dans leur ligne de commande ne contient plus d'options, et ne doit donc plus être interprété. Il suffit donc de faire précéder le nom du fichier par deux tirets pour arrêter l'interprétation des options. Par exemple, pour afficher les informations relatives à un fichier nommé « -l », il faudrait utiliser la commande suivante :

ls -l -- -l

Enfin, sachez que le caractère '~' représente le répertoire personnel de l'utilisateur courant. Il est donc très facile d'accéder aux fichiers et aux répertoires de son répertoire personnel (situé, rappelons-le, dans le répertoire /home/). Par exemple, pour lister le contenu de son répertoire personnel, on utilisera la commande suivante :

ls -l ~

6. Opérations de base sur les fichiers

Vous aurez sans doute à afficher le contenu d'un fichier. Pour cela, la commande la plus appropriée est certainement la commande **less** :

less fichier

Cette commande affiche le contenu du fichier et vous permet de le faire défiler avec les flèches du curseur. Lorsque vous désirez terminer la visualisation, il suffit de taper la touche q (pour « quitter » **less**). Pour information, le nom de la commande **less** provient d'un trait d'humour sur une commande Unix plus classique, la commande **more**. Cette commande effectue à peu près le même travail que **less**, mais elle n'affiche le texte que page par page. Pour passer à la page suivante, il faut appuyer sur la barre d'espace. Quant à l'origine du nom de la commande **more**, c'est qu'elle affiche le mot « more » au bas de l'écran pour indiquer qu'il y a encore du texte à visualiser, et qu'il faut appuyer sur la barre d'espace pour lire la suite.

La commande **less** permet également d'effectuer une recherche dans le fichier en cours d'édition. Pour cela, il suffit de taper une commande de recherche de **less**. Cette commande commence par une barre oblique, suivie du texte à chercher. Par exemple, pour rechercher la chaîne de caractères « local » dans un fichier en cours de visualisation avec **less**, il suffit de taper :

/local

Lorsque vous voudrez rechercher l'occurrence suivante du motif de recherche, vous pourrez appuyer sur la touche n (pour « Next » en anglais). Pour rechercher l'occurrence précédente, il suffit de taper la touche N (en majuscule, cette fois).

Il est encore plus probable que vous aurez à éditer un fichier. Cette opération peut se faire relativement facilement grâce à un éditeur simplifié, **vi**. Cet éditeur n'est pas franchement ce qui se fait de plus convivial, cependant, il existe sur toutes les plates-formes Unix d'une part, et il est suffisamment léger pour pouvoir fonctionner sur un système minimal. Il est donc recommandé de savoir se servir de **vi**, ne serait-ce que dans le cas où votre système ne serait pas complètement fonctionnel. En clair, quand tout va mal, on peut compter sur **vi** ! **vi** sera décrit plus loin dans la chapitre 8, car il dispose d'un grand nombre de commandes et il ne serait pas opportun de les décrire ici.

En général, la création d'un fichier se fait avec **vi**, bien que d'autres commandes puissent créer des fichiers. En revanche, pour supprimer un fichier, il n'existe qu'une seule commande :

rm chemin

où chemin est le chemin complet permettant d'accéder au fichier à supprimer. Il est possible de spécifier plusieurs fichiers à la commande **rm**. Dans ce cas, ils seront tous supprimés. **rm** est également capable de

supprimer tous les fichiers d'un répertoire, ainsi que ses sous-répertoires. Dans ce cas, elle détruit toute une branche de l'arborescence du système de fichiers. Pour cela, il suffit d'utiliser l'option -r (pour « récursif ») avant le chemin du répertoire à supprimer.

Attention ! : La commande **rm** ne demande aucune confirmation avant de supprimer les fichiers ! D'autre part, les fichiers supprimés sont irrémédiablement perdus (il n'y a pas de commande « undelete » ou autre commande similaire). Vérifiez donc bien ce que vous avez tapé avant de valider une commande **rm** (surtout si vous êtes sous le compte root). Il peut être judicieux de forcer la commande **rm** à demander confirmation avant la suppression des fichiers, à l'aide de son option -i. On pourra pour cela définir un alias « rm -i » pour la commande **rm** dans le fichier d'initialisation du shell (c'est-à-dire le fichier .bashrc pour le shell bash).

La copie d'un fichier se fait avec la commande **cp**, dont la syntaxe est donnée ci-dessous :

cp fichiers répertoire

où fichiers est la liste des fichiers à copier, et répertoire est le répertoire destination dans lequel ces fichiers doivent être copiés.

Enfin, le déplacement des fichiers se fait avec la commande **mv**, comme indiqué ci-dessous :

mv source destination
où source est le nom du fichier source et destination est le nom du répertoire destination. Notez que **mv** est une commande très puissante, puisqu'elle permet également de déplacer des répertoires et de renommer des fichiers et des répertoires. Pour renommer un fichier ou un répertoire, il suffit d'indiquer le nouveau nom de ce fichier ou de ce répertoire à la place de destination.

7. Autres commandes utiles

Pour terminer ce petit cours d'Unix, nous allons décrire quelques-unes des autres commandes d'Unix parmi les plus utiles. Elles sont utilisées moins souvent que les commandes vues précédemment, mais vous apprendrez certainement très vite à vous en servir, car elles sont très pratiques.

7.1. Gestion des liens

La commande pour créer un lien est **ln**. Cette commande utilise la syntaxe suivante :

ln [-s] source lien

où source est le nom du fichier ou du répertoire source auquel le lien doit se référer, et lien est le nom du lien. L'option -s permet de créer un lien symbolique. Par défaut, ce sont des liens physiques qui sont créés. Rappelons qu'il est impossible de créer des liens physiques sur des

répertoires.

Lorsqu'on liste des fichiers, on peut demander l'affichage d'informations complémentaires sur les liens. Pour cela, il suffit d'utiliser l'option -l de la commande **ls**. Ainsi, la commande suivante :

ls -l lien
: permet d'afficher les informations sur le lien lien, et en particulier le fichier ou le répertoire cible de ce lien.

La suppression des liens se fait exactement comme celle d'un fichier. La destination n'est pas affectée en général, sauf si le lien est un lien physique et constitue la dernière référence au fichier pointé par le lien.

Les liens symboliques n'ont pas de droits d'accès ni de propriétaires, les informations de sécurité de la cible sont utilisées lorsqu'on accède au lien.

7.2. Recherche de fichiers

Il vous sera sans doute nécessaire de rechercher des fichiers selon un critère donné dans toute une arborescence de répertoires. Pour cela, vous utiliserez la commande **find**. Cette commande est très puissante, mais dispose d'une syntaxe assez compliquée :

find répertoire -name nom -print
: où répertoire est le répertoire à partir duquel la recherche doit commencer et nom est le nom du fichier à rechercher. Ce nom peut contenir des caractères génériques du shell, mais dans ce cas doit être placé entre guillemets afin d'éviter que ce dernier ne les interprète.

find accepte d'autres options de recherche que le nom (partie « -name » de la ligne de commande), et peut effectuer d'autres actions que l'affichage du chemin des fichiers trouvés (partie « -print »). Consultez les pages de manuel pour plus d'informations à ce sujet.

7.3. Recherche d'un texte dans un fichier

La recherche d'une chaîne de caractères dans un ou plusieurs fichiers peut se faire à l'aide de la commande **grep**. Cette commande prend en premier paramètre le texte à rechercher, puis la liste des fichiers dans lequel ce texte doit être trouvé :

grep texte fichiers

Le texte peut être placé entre guillemets si nécessaire (en particulier, s'il contient des espaces ou des caractères interprétés par le shell, comme * et ?). **grep** accepte un grand nombre d'options, qui ne seront pas décrites ici. Consulter les pages de manuel pour plus d'information à ce sujet.

7.4. Remplacement de texte dans les fichiers

Le remplacement de texte dans un fichier peut être effectué de manière automatique, c'est-à-dire sans avoir à ouvrir le fichier dans un éditeur, grâce à la commande **sed** (abréviation de l'anglais « Stream Editor »). Cette commande est en fait un utilitaire de manipulation de flux de données, qui permet d'effectuer des traitements plus généraux que le simple remplacement de texte, mais c'est malgré tout pour cette opération qu'elle reste la plus utilisée.

sed peut travailler à la volée sur un flux de données textuelles, que ce flux provienne de l'entrée standard ou d'un fichier. Par défaut, il écrit le résultat de son travail sur le flux de sortie standard. Les opérations qu'il doit effectuer sur le flux de données peuvent être spécifiées de différentes manières, soit en fournissant un fichier script à l'aide de l'option -f, soit directement sur la ligne de commande, avec l'option -e. La syntaxe utilisée pour appeler **sed** est donc typiquement la suivante :

sed -e "commandes" fichier > résultat

ou :

sed -f script fichier > résultat

où fichier est le fichier sur lequel **sed** doit travailler, et résultat est le fichier devant recevoir le flux de données modifiées. Notez que cette commande utilise une redirection du flux de sortie standard dans un fichier.

sed peut effectuer un grand nombre de commandes différentes et est réellement un outil très puissant. Cependant, nous ne verrons ici que la commande qui permet d'effectuer un remplacement de texte. Cette commande utilise la syntaxe suivante :

s/texte/remplacement/options

où texte est le texte à rechercher, remplacement est le texte de remplacement, et options est un jeu d'options exprimant la manière dont le remplacement doit être fait. Les options sont spécifiées à l'aide de simple caractères, les plus utiles étant sans doute g, qui permet d'effectuer un remplacement global (au lieu de ne remplacer que la première occurrence du texte rencontrée dans chaque ligne), et I, qui permet d'effectuer une recherche sans tenir compte de la casse des caractères.

Par exemple, la ligne de commande suivante :

sed -e "s/bonjour/bonsoir/g" test.txt > modif.txt

permet de remplacer toutes les occurrences de la chaîne de caractères « bonjour » par la chaîne de caractères « bonsoir » dans le texte du fichier test.txt, et d'enregistrer le résultat dans le fichier modif.txt.

Note : Il ne faut pas utiliser le même nom de fichier pour le fichier source et le fichier de résultat. En effet, **sed** lit le fichier source à la volée, et effectuer une redirection sur ce fichier pendant son traitement provoquerait la perte irrémédiable de son contenu. Pour résoudre ce problème, on pourra utiliser un nom de fichier temporaire, et écraser le fichier original par ce fichier une fois la commande **sed** exécutée.

7.5. Compression et décompression des fichiers

Linux fournit un grand nombre de programmes de compression de fichiers. Le meilleur est sans doute **bzip2**, et le plus compatible sans doute **compress**. Cependant, le plus utilisé et le plus courant, surtout pour la distribution des sources, reste incontestablement **gzip**. Nous allons décrire brièvement comment compresser et décompresser des fichiers avec **gzip** et **bzip2** dans ce paragraphe.

La compression d'un fichier se fait de manière élémentaire :

gzip fichier
: où fichier est le fichier à compresser. Après avoir effectué son travail, **gzip** renomme le fichier compressé en « fichier.gz ». La compression d'un fichier avec **bzip2** utilise exactement la même syntaxe, à ceci près qu'il faut remplacer gzip par bzip2. De plus, le nom du fichier compressé porte l'extension .bz2 au lieu de .gz. Le fichier obtenu est donc nommé « fichier.bz2 ».

La décompression d'un fichier se fait à l'aide de la commande suivante :

gunzip fichier.gz
: ou

bunzip2 fichier.bz2
: selon qu'il a été compressé avec **gzip** ou **bzip2**. Après décompression, l'extension complémentaire .gz ou .bz2 est supprimée du nom de fichier.

7.6. Archivage de fichiers

L'archivage de fichiers se fait classiquement sous Unix avec le programme **tar** (abréviation de l'anglais « Tape ARchiver »). Ce programme permet simplement de regrouper tous les fichiers qu'il doit archiver dans un seul fichier structuré en blocs. Il a été initialement écrit pour permettre des archivages sur bandes ou sur tout autre périphérique de stockage de masse, mais il est également utilisé pour créer des fichiers archives contenant toute une arborescence.

La syntaxe de **tar** est très simple :

tar options archive [fichiers]
: où options sont les options qui indiquent l'opération à effectuer et comment elle doit être réalisée, archive est le nom de l'archive qui doit être créée ou le nom du fichier de périphérique du périphérique

d'archivage, et fichiers est la liste des fichiers à archiver.

Les options de **tar** que vous utiliserez le plus souvent sont les suivantes :

- cvf pour créer une archive ;
- tvf pour lister le contenu d'une archive ;
- xvf pour restaurer le contenu d'une archive.

Par exemple, pour archiver le contenu du répertoire courant dans le fichier archive.tar, vous utiliserez la ligne de commande suivante :

tar cvf archive.tar *

De plus, pour extraire le contenu de l'archive archive.tar, vous utiliserez la commande suivante :

tar xvf archive.tar

Note : L'option z permet d'effectuer une compression des données archivées ou une décompression des données restaurées à la volée. **tar** utilise **gzip** et **gunzip** pour la compression et la décompression. De même, l'option j permet de compresser l'archive à la volée avec **bzip2**.

Si l'on utilise un signe négatif ('-') à la place du nom de l'archive, **tar** enverra le résultat de la compression vers la sortie standard.

7.7. Passage en mode superviseur

Si vous êtes prudent, vous avez sans doute créé un compte utilisateur juste après avoir installé votre système de base, et vous ne travaillez plus que dans ce compte. Cette technique est prudente, cependant, elle pose un problème évident : vous ne pouvez pas y faire votre travail d'administrateur. C'est pour cela que la commande **su** a été créée. Cette commande permet de changer son identité dans le système :

su [utilisateur]

où utilisateur est l'utilisateur dont on veut prendre l'identité. Par défaut, si aucun utilisateur n'est spécifié, le changement d'identité se fait vers l'utilisateur root. Bien entendu, il va de soi que la commande su demande le mot de passe avant d'obtempérer...

7.8. Changement des droits des fichiers, du propriétaire et du groupe

7.8.1. Changement de propriétaire et de groupe

Le changement de propriétaire d'un fichier ne peut être réalisé que par l'administrateur du système. Cette opération se fait à l'aide de la commande suivante :

`chown utilisateur fichier`

où `utilisateur` est le nom de l'utilisateur qui doit devenir propriétaire du fichier, et `fichier` est le fichier devant changer de propriétaire.

Le changement de groupe peut être réalisé par n'importe quel utilisateur, mais on ne peut donner un fichier qu'à l'un des groupes dont on est membre. Cette opération se fait à l'aide de la commande suivante :

`chgrp groupe fichier`

où `groupe` est le nom du groupe qui doit être affecté au fichier, et `fichier` est le fichier devant changer de groupe. Bien entendu, l'administrateur peut affecter un fichier à n'importe quel groupe d'utilisateur.

7.8.2. Modification des droits Unix sur les fichiers

La commande permettant de changer les droits d'un fichier ou d'un répertoire est la suivante :

chmod droits fichier

où fichier est le fichier ou le répertoire dont on désire changer les droits, et droits est une chaîne de caractères permettant de spécifier les nouveaux droits. Cette chaîne commence par une lettre indiquant le groupe d'utilisateurs auquel le droit doit être appliqué, d'un caractère + ou - indiquant si le droit doit être ajouté ou supprimé, et d'une lettre indiquant le droit que l'on est en train de manipuler. La première lettre peut prendre les valeurs suivantes :

- u pour le champ « utilisateur », c'est-à-dire le propriétaire du fichier ;
- g pour le champ « groupe », c'est-à-dire tous les utilisateurs faisant partie du groupe du fichier ;
- o pour le champ « other », c'est-à-dire pour tous les utilisateurs qui ne sont ni propriétaires, ni membres du groupe du fichier ;
- a pour tous les champs sans distinction, donc pour tous les utilisateurs.

Les droits sont identifiés par l'un des caractères suivants :

- r pour le droit de lecture ;
- w pour le droit d'écriture ;
- x pour le droit d'exécution ;
- s pour les bits setuid et setguid ;

- t pour le bit sticky.

Ainsi, la commande suivante :

chmod g+w exemple
permet de donner le droit d'écriture sur le fichier exemple à tous les membres du groupe auquel ce fichier appartient.

7.8.3. Utilisation des ACLs

Si l'on veut donner des droits à un utilisateur ou un groupe particulier, on pourra définir une ACL (« Access Control List ») sur le fichier ou le répertoire, et affecter les droits unitairement. Ceci se fait simplement avec la commande **setfacl**, de la manière suivante :

setfacl -m ACL fichier
où ACL est l'ACL à affecter au fichier ou au répertoire fichier. Les ACLs sont constitués d'une liste d'entrées nommées des ACE (« Access Control Entries »). Les ACEs sont séparées par des virgules et définissent chacune un droit. Chacun de ces droits doit être spécifié de manière complète, en précisant la classe d'utilisateur sur lequel il porte avec un mot-clé (user pour l'utilisateur propriétaire, group pour les utilisateurs du groupe propriétaire, ou other pour les autres utilisateurs), le nom de l'utilisateur ou du groupe si nécessaire, et les droits affectés à cet utilisateur ou à ce groupe. Tous ces paramètres doivent être séparés par deux points (caractère ':'). Par exemple, pour ajouter les droits d'écriture à l'utilisateur alfred sur le fichier exemple, vous pourrez utiliser la commande suivante :
setfacl -m user:hamid:w exemple

Si l'on ne spécifie aucun utilisateur avec la classe d'utilisateurs user dans une entrée, cette entrée se réfère automatiquement à l'utilisateur propriétaire du fichier ou du répertoire. De même, si l'on ne spécifie aucun groupe avec la classe d'utilisateurs group, l'entrée se réfère au groupe auquel le fichier appartient. Fixer ces entrées d'ACL sur un fichier avec ces deux syntaxes revient donc exactement à utiliser **chmod** avec les droits Unix classiques (à un détail près que l'on verra ci-dessous pour le groupe).

Les droits complets d'un fichier ou d'un répertoire peuvent être consultés avec la commande **getfacl**. Cette commande affiche en commentaire les informations sur l'objet auquel elle s'applique, à savoir son nom, son propriétaire et son groupe, à la suite d'un dièze (caractère '#'). Suivent toutes les ACEs affectées à cet objet. Les droits Unix classiques sont lisibles directement avec les entrées user::, group:: et other:: respectivement pour l'utilisateur, les utilisateurs du groupe et les autres utilisateurs. Par exemple, si l'on affiche les ACLs associées au fichier exemple après la commande précédente, on obtient ceci :

```
# file: exemple
# owner: rachid
# group: users
user::rw-
user:hamid:-w-
group::r--
mask::rw-
other::r--
```

Dès qu'une ACL nominative a été attribuée à un fichier, une ACL spéciale est créée automatiquement. Il s'agit de l'ACL mask, qui, comme son nom l'indique, définit un masque de droits complémentaires pour tous les utilisateurs et les groupes ajoutés nominativement à l'ACL. Pour ces utilisateurs, les droits effectivement accordés sont leurs droits respectifs, restreints aux droits présent dans le masque. Le masque permet donc de restreindre les droits de tous les utilisateurs de la classe group, au sens large. Par exemple, si le masque contient les droits de lecture et d'écriture, et que l'utilisateur alfred dispose des droits de lecture et d'exécution sur le fichier, ses droits effectifs seront uniquement la lecture. Les droits effectifs sont indiqués en commentaire par **getfacl** pour les utilisateurs et les groupes s'ils ne sont pas égaux aux droits indiqués dans leurs entrées respectives.

Dès qu'il est défini, le masque remplace l'entrée du groupe du fichier pour les commandes classiques. Ainsi, dès qu'un masque est défini dans l'ACL d'un fichier, les changements de droits sur le groupe effectués avec la commande **chmod** ne modifient plus que le champ de masque. Cela peut surprendre dans certaines situations. Par exemple, si l'entrée de l'ACL décrivant les droits du groupe du fichier ne donne aucun droit, les utilisateurs de ce groupe n'auront toujours aucun droit même après un **chmod g+rwx** sur le fichier. En effet, cette dernière commande ne modifie que le masque. Il est donc nécessaire d'ajouter les droits sur le groupe du fichier explicitement avec **setfacl**, de la manière suivante :

```
chmod g+rwx exemple
setfacl -m group::rwx exemple
```

La première commande modifie les droits sur le masque, et la deuxième ajoute les droits pour tous les utilisateurs du groupe.

Notez bien que le masque ne s'applique pas pour la détermination des droits du propriétaire et des utilisateurs de la classe other. Par ailleurs, lorsque l'on affiche les droits étendus d'un fichier avec l'option -l de la commande **ls**, les droits affichés pour le groupe du fichier sont les droits définis dans le masque lui-même (même si aucune entrée d'ACL ne donne complètement ces droits à un utilisateur ou à un groupe). Cela permet donc de voir directement les droits les plus forts qui *pourraient* être attribués sur ce fichier, ce qui est cohérent avec ce

qu'attendent les outils Unix classiques.

Il est possible de supprimer toutes les entrées de l'ACL d'un fichier avec l'option -b de **setfacl**. Pour supprimer une entrée spécifique de l'ACL, vous devrez utiliser l'option -x. Cette dernière option ne permet pas de supprimer les entrées génériques pour le propriétaire et le groupe du fichier, ni celles des autres utilisateurs. Par exemple, pour retirer toutes les entrées définies précédemment sur le fichier exemple, on utilisera la commande suivante :

setfacl -b exemple

Vous pourrez toutefois constater que **getfacl** continue d'afficher les entrées génériques « user:: », « group:: » et « other:: ».

Les répertoires disposent, en plus de leur ACL normale, d'une ACL par défaut. L'ACL par défaut d'un répertoire est celle qui est appliquée à tout nouveau fichier ou répertoire créés dans ce répertoire. De plus, les sous-répertoires héritent de l'ACL par défaut de leur parent. Les ACLs par défaut sont modifiables avec l'option -d de **setfacl** et sont affichées avec le préfixe default: par **getfacl**.

Note : Notez bien que de nombreux outils Unix classiques ne gèrent pas correctement la notion d'ACL (en particulier les gestionnaires de fichiers graphiques). La copie ou la modification d'un fichier peut donc provoquer la perte de son ACL, modifiant ainsi les droits des utilisateurs sur ce fichier. Les ACLs doivent donc être utilisées avec parcimonie et leur manipulation entourée du plus grand soin.

8. vi, l'éditeur de fichiers de base

Vous serez obligé, lorsque vous effectuerez la configuration de votre système, d'éditer les fichiers de configuration (classiquement, ces fichiers sont placés dans le répertoire /etc/). Ces modifications peuvent être réalisées avec n'importe quel éditeur a priori, et il est même conseillé d'utiliser votre éditeur favori. Cependant, il faut savoir se servir de vi, parce que c'est le seul éditeur qui sera toujours installé, et qui fonctionnera en toutes circonstances. Le prix à payer pour cette fiabilité est un nombre restreint de fonctionnalités. En fait, vi est très puissant, mais il ne s'embarrasse pas de superflu, ce qui en fait certainement l'éditeur le moins convivial du monde. Ce paragraphe vous donnera la liste des principales commandes de vi. Cette liste ne sera toutefois pas exhaustive, car vous n'utiliserez certainement pas vi dans la vie courante.

Pour éditer un fichier avec vi, il suffit de passer le nom de ce fichier en ligne de

commande :

vi fichier

Il est possible de passer plusieurs fichiers dans la ligne de commande, et vi les éditera les uns après les autres. Cependant, il faut savoir que vi ne permet de travailler que sur deux fichiers à la fois, et qu'il n'est pas facile de passer de l'un à l'autre. Par conséquent, il est conseillé de n'éditer qu'un seul fichier à la fois.

vi est un éditeur qui fonctionne dans plusieurs modes différents : le *mode d'édition*, dans lequel le texte peut être modifié, le *mode de commande*, dans lequel des commandes particulières peuvent être données, et le *mode de visualisation*, dans lequel le fichier ne peut être que visualisé. Par défaut, vi est en mode de visualisation, et il faut utiliser une commande d'édition pour passer en mode d'édition. Quand on est en mode d'édition, on peut revenir au mode de visualisation en appuyant sur la touche Echap (ou Esc, selon votre clavier). Cette touche a aussi une signification dans le mode de commande : elle annule la saisie de la commande en cours. Par conséquent, lorsqu'on est perdu et que l'on ne sait plus dans quel mode on se trouve (ce qui arrive fatalement à un moment donné), il suffit d'appuyer sur cette touche. On sait alors qu'on se trouve en mode de visualisation.

Le déplacement du curseur en mode de visualisation se fait avec les touches du curseur. Cependant, si votre clavier n'est pas bien configuré, ces touches peuvent ne pas fonctionner. C'est pour cette raison que vi fournit un jeu de touches alternatif :

- la touche h permet de déplacer le curseur vers la gauche ;
- la touche l permet de déplacer le curseur vers la droite ;
- la touche j permet de déplacer le curseur vers le bas ;
- la touche k permet de déplacer le curseur vers le haut.

> Le curseur est bien entendu déplacé automatiquement lors de la saisie du texte en mode d'édition.

Le passage en mode d'édition peut se faire avec l'une des commandes suivantes :

- la touche i permet de passer en mode d'insertion (le texte saisi s'insère avant le caractère sur lequel le curseur est positionné) ;
- la touche a permet de passer en mode d'ajout de caractères (le texte saisi s'insère après le caractère sur lequel le curseur est positionné) ;
- la touche A permet de placer le curseur en fin de ligne et de passer en mode d'ajout de caractères ;
- la touche o permet de créer une nouvelle ligne après la ligne où se trouve le curseur et de passer en mode d'édition sur cette nouvelle ligne ;
- la touche O permet de créer une nouvelle ligne avant la ligne où se trouve le curseur et de passer en mode d'édition sur cette nouvelle ligne.

La création d'une nouvelle ligne peut donc être faite avec les commandes o et O, mais il est possible de couper une ligne en deux, ou de passer à la ligne

simplement en tapant sur la touche Entrée en mode d'édition. Inversement, la commande J permet de supprimer un saut de ligne en fin de ligne et de placer ainsi le texte de la ligne suivante à la suite du texte de la ligne courante.

La suppression d'un caractère se fait avec la touche Suppr (ou Del, selon le clavier) ou la touche de retour arrière (dite touche Backspace). Cependant, encore une fois, vi fournit un jeu de touches alternatif permettant de travailler avec un clavier mal configuré :

- la commande x permet d'effacer le caractère situé sous le curseur ;
- la commande dd permet d'effacer la ligne où se trouve le curseur ;
- la commande dw permet d'effacer le mot où se trouve le curseur.

  Le texte qui a été supprimé est placé dans ce que l'on appelle un *buffer*. Le contenu du buffer peut être inséré à n'importe quel endroit du fichier grâce à la commande p. Ainsi, il est possible de faire un couper/coller en effaçant la ligne désirée et en appuyant sur la touche p à l'emplacement destination.

La commande u permet d'annuler la dernière opération effectuée, et la commande U permet de la ré-exécuter.

La commande yy permet de copier la ligne courante dans le buffer. Cette commande est donc utilisée pour effectuer des copier/coller, en combinaison avec la commande p.

Les commandes de vi peuvent être répétées un certain nombre de fois, en spécifiant ce nombre avant de les écrire. Ainsi, pour supprimer 3 lignes, il suffira de taper la commande suivante :

3dd

Dans ce cas, ces trois lignes sont également placées dans le buffer. La même technique peut être utilisée pour copier/coller plusieurs lignes en une seule opération.

Enfin, vi accepte un certain nombre de commandes générales lorsqu'il est en mode de commande. Ce mode est activé dès que l'on appuie sur la touche deux points (':') dans le mode de visualisation. Les commandes générales les plus utiles sont décrites ci-dessous :

- la commande :q permet de quitter vi. Si le fichier en cours d'édition a été modifié, vi refusera de se terminer sans l'enregistrer. Si l'on veut malgré tout sortir sans l'enregistrer, il faudra utiliser la commande :q! ;
- la commande :w permet d'enregistrer le fichier courant. Pour enregistrer ce fichier et quitter vi, la commande :wq peut être utilisée ;
- la commande :help sujet permet d'obtenir de l'aide sur le sujet « sujet » ;
- la commande :!commande permet d'exécuter la commande du shell « commande ». Cela peut être pratique pour effectuer une opération dans le shell sans avoir à quitter vi. Cela dit, il sera sans doute plus efficace

d'utiliser un autre terminal virtuel.

Comme vous l'avez constaté, vi est réellement une horreur à utiliser. Malgré tout, il permet de faire tout ce dont on a besoin pour éditer un fichier. Il dispose même de puissantes fonctionnalités que même les traitements de texte évolués ne sont pas capables de faire. Elles ne seront cependant pas décrites ici, car cela dépasserait le cadre de la simple installation de Linux. Vous pourrez toujours consulter l'aide de vi pour de plus amples informations.