

Chapitre 02 : Utiliser le Shell de Linux

2.1 Interface de ligne de commande

La plupart des systèmes d'exploitation grand public sont conçus pour protéger l'utilisateur des « tenants et aboutissants » de la CLI. La communauté Linux est différente dans la mesure où elle célèbre positivement la CLI pour sa puissance, sa rapidité et sa capacité à accomplir une vaste gamme de tâches avec une seule instruction de ligne de commande.

Lorsqu'un utilisateur rencontre la CLI pour la première fois, il peut trouver cela difficile car cela nécessite de mémoriser une quantité vertigineuse de commandes et de leurs options. Cependant, une fois qu'un utilisateur a appris la structure d'utilisation des commandes, où se trouvent les fichiers et répertoires nécessaires et comment naviguer dans la hiérarchie d'un système de fichiers, il peut être extrêmement productif. Cette fonctionnalité offre un contrôle plus précis, une plus grande vitesse et la possibilité d'automatiser les tâches plus facilement via des scripts.

De plus, en apprenant la CLI, un utilisateur peut facilement être productif presque instantanément sur N'IMPORTE QUELLE version ou distribution de Linux, réduisant ainsi le temps nécessaire pour se familiariser avec un système en raison des variations d'une interface graphique.

2.2 Commandes

Qu'est-ce qu'une commande ? La réponse la plus simple est qu'une commande est un programme logiciel qui, lorsqu'il est exécuté sur la ligne de commande, exécute une action sur l'ordinateur.

Lorsque vous envisagez une commande utilisant cette définition, vous réfléchissez réellement à ce qui se passe lorsque vous exécutez une commande. Lorsque vous tapez une commande, un processus est exécuté par le système d'exploitation qui peut lire les entrées, manipuler les données et produire une sortie. De ce point de vue, une commande exécute un processus sur le système d'exploitation, qui amène ensuite l'ordinateur à effectuer une tâche.

Cependant, il existe une autre façon de voir ce qu'est une commande : regardez sa source. La source est l'endroit d'où « vient » la commande et il existe plusieurs sources différentes de commandes dans le shell de votre CLI :

Commandes internes : également appelées commandes intégrées, ces commandes sont intégrées au shell lui-même. Un bon exemple est la commande **cd** (changer de répertoire) car elle fait partie du shell Bash. Lorsqu'un utilisateur tape la commande **cd**, le shell Bash est déjà en cours d'exécution et sait

comment interpréter cette commande, ne nécessitant le démarrage d'aucun programme supplémentaire.

Commandes externes : ces commandes sont stockées dans des fichiers recherchés par le shell. Si vous tapez la commande **ls**, le shell recherche dans une liste prédéterminée de répertoires pour essayer de trouver un fichier nommé **ls** qu'il peut exécuter. Ces commandes peuvent également être exécutées en tapant le chemin complet de la commande.

Alias : un alias peut remplacer une commande intégrée, une fonction ou une commande trouvée dans un fichier. Les alias peuvent être utiles pour créer de nouvelles commandes à partir de fonctions et de commandes existantes.

Fonctions : les fonctions peuvent également être créées à l'aide de commandes existantes, pour créer de nouvelles commandes, remplacer les commandes intégrées au shell ou les commandes stockées dans des fichiers. Les alias et les fonctions sont normalement chargés à partir des fichiers d'initialisation au premier démarrage du shell, comme indiqué plus loin dans cette section.

2.2.1 Commandes externes

Les commandes stockées dans des fichiers peuvent se présenter sous plusieurs formes que vous devez connaître. La plupart des commandes sont écrites dans le langage de programmation C, qui est initialement stocké dans un fichier texte lisible par l'homme. Ces fichiers source texte sont ensuite compilés en fichiers binaires lisibles par ordinateur, qui sont ensuite distribués sous forme de fichiers de commandes.

Les utilisateurs qui souhaitent voir le code source d'un logiciel compilé sous licence GPL peuvent le trouver sur les sites d'où il provient. Le code sous licence GPL oblige également les distributeurs des binaires compilés, tels que RedHat et Debian, à rendre le code source disponible. On le trouve souvent dans les référentiels des distributeurs.

Note

Il est possible de visualiser les logiciels disponibles, les programmes binaires qui peuvent être installés directement en ligne de commande. Tapez la commande suivante dans le terminal pour afficher les **packages sources** disponibles (code source pouvant être modifié avant d'être compilé en programmes binaires) pour la collection de compilateurs GNU :

```
sysadmin@localhost:~$ apt-cache search gcc | grep source
gcc-4.8-source - Source of the GNU Compiler Collection
gcc-5-source - Source of the GNU Compiler Collection
gcc-6-source - Source of the GNU Compiler Collection
gcc-7-source - Source of the GNU Compiler Collection
gcc-8-source - Source of the GNU Compiler Collection
gcc-arm-none-eabi-source - GCC cross compiler for ARM Cortex
```

La commande **apt-cache** nous permet d'afficher les informations du cache de la base de données **APT**. Il est couramment utilisé pour trouver des informations sur les programmes que vous souhaitez installer et les composants requis pour les faire fonctionner.

Bien qu'il existe un très grand nombre de programmes gratuits et open source disponibles, bien souvent, le code binaire dont vous aurez besoin en tant qu'administrateur Linux n'existera pas pour la distribution particulière que vous utilisez. Étant donné que les licences open source vous donnent accès au code de ces programmes, l'une de vos tâches consistera à compiler et parfois à modifier ce code en programmes exécutables pouvant être installés sur les systèmes que vous gérez. La **Free Software Foundation (FSF)** distribue la **GNU Compiler Collection (GCC)** pour faciliter ce processus. Le GCC fournit un système de compilation (les programmes spéciaux utilisés pour convertir le code source en programmes binaires utilisables) avec des interfaces pour de nombreux langages de programmation différents. En fait, la FSF ne limite pas ces outils à Linux. Il existe des versions de GCC qui fonctionnent sous Windows, MacOS et de nombreux autres systèmes, y compris des environnements de microcontrôleurs spécifiques.

La gestion des packages Linux sera abordée plus en détail plus tard dans le cours.

2.2.2 Alias

Un alias peut être utilisé pour mapper des commandes plus longues à des séquences de touches plus courtes. Lorsque le shell voit un alias en cours d'exécution, il remplace la séquence la plus longue avant de procéder à l'interprétation des commandes.

Pour déterminer quels alias sont définis sur le shell actuel, utilisez la commande **alias** :

```
sysadmin@localhost:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

Les alias des exemples précédents ont été créés par des fichiers d'initialisation. Ces fichiers sont conçus pour rendre automatique le processus de création d'alias.

De nouveaux alias peuvent être créés en utilisant le format suivant, où **nom** est le **name** à donner à l'alias et **command** est la commande à exécuter lors de l'exécution de l'alias.

```
alias name=command
```

Par exemple, la commande **cal 2030** affiche le calendrier de l'année 2030. Supposons que vous exécutiez souvent cette commande. Au lieu d'exécuter la commande complète à chaque fois, vous pouvez créer un alias appelé **mycal** et exécuter l'alias, comme illustré dans le graphique suivant :

```
sysadmin@localhost:~$ alias mycal="cal 2030"
sysadmin@localhost:~$ mycal

                2030
    January          February          March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
    1  2  3  4  5              1  2              1  2
  6  7  8  9 10 11 12    3  4  5  6  7  8  9    3  4  5  6  7  8  9
 13 14 15 16 17 18 19   10 11 12 13 14 15 16   10 11 12 13 14 15 16
 20 21 22 23 24 25 26   17 18 19 20 21 22 23   17 18 19 20 21 22 23
 27 28 29 30 31         24 25 26 27 28         24 25 26 27 28 29 30
                                     31

    April          May          June
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6              1  2  3  4              1
  7  8  9 10 11 12 13    5  6  7  8  9 10 11    2  3  4  5  6  7  8
 14 15 16 17 18 19 20   12 13 14 15 16 17 18    9 10 11 12 13 14 15
 21 22 23 24 25 26 27   19 20 21 22 23 24 25   16 17 18 19 20 21 22
 28 29 30              26 27 28 29 30 31       23 24 25 26 27 28 29
                                     30

    October          November          December
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
    1  2  3  4  5              1  2              1  2  3  4  5  6  7
  6  7  8  9 10 11 12    3  4  5  6  7  8  9    8  9 10 11 12 13 14
```

Les alias créés de cette manière ne persistent que lorsque le shell est ouvert. Une fois le shell fermé, les nouveaux alias sont perdus. De plus, chaque shell a ses propres alias, donc les alias créés dans un shell ne seront pas disponibles dans un nouveau shell ouvert.

2.3 Syntaxe de commande de base

Pour exécuter une commande, la première étape consiste à saisir le nom de la commande. Cliquez dans le terminal. Tapez **ls** et appuyez sur Entrée. Le résultat devrait ressembler à l'exemple ci-dessous :

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Template
```

Note

À elle seule, la commande `ls` répertorie les fichiers et répertoires contenus dans le répertoire de travail actuel. À ce stade, vous ne devriez pas trop vous soucier du résultat de la commande, mais plutôt vous concentrer sur la compréhension de la façon de formater et d'exécuter les commandes.

La commande `ls` sera abordée plus en détail plus tard dans le cours.

De nombreuses commandes peuvent être utilisées seules sans autre intervention. Certaines commandes nécessitent des entrées supplémentaires pour s'exécuter correctement. Cette entrée supplémentaire se présente sous deux formes : **options** et **arguments**. Les commandes suivent généralement un modèle de syntaxe simple :

```
command [options...] [arguments...]
```

Lors de la saisie d'une commande à exécuter, la première étape consiste à saisir le nom de la commande. Le nom de la commande est souvent basé sur ce que fait la commande ou sur ce que le développeur qui a créé la commande pense décrire le mieux la fonction de la commande.

Par exemple, la commande `ls` affiche une liste d'informations sur les fichiers. Associer le nom de la commande à quelque chose de mnémonique pour ce qu'elle fait peut vous aider à vous souvenir plus facilement des commandes.

Gardez à l'esprit que chaque partie de la commande est normalement sensible à la casse, donc **LS** est incorrect et échouera, mais **ls** est correct et réussira.

2.3.1 Spécification des arguments

```
command [options] [arguments]
```

Un argument peut être utilisé pour spécifier quelque chose sur lequel la commande doit agir. Suite à une commande, tous les arguments souhaités sont autorisés ou requis en fonction de la commande. Par exemple, la commande **touch** est utilisée pour créer des fichiers vides ou mettre à jour l'horodatage des fichiers existants. Il nécessite au moins un argument pour spécifier le nom du fichier sur lequel agir.

```
touch FILE...
```

```
sysadmin@localhost:~$ touch newfile
```

La commande ls, quant à elle, permet de spécifier un chemin et/ou un nom de fichier comme argument, mais ce n'est pas obligatoire.

```
ls [FILE]...
```

Un exemple de scénario dans lequel un argument est autorisé mais pas obligatoire est l'utilisation de la commande ls. Si la commande ls est utilisée sans argument, elle listera le contenu du répertoire courant :

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Template
```

Si la commande ls reçoit le nom d'un répertoire comme argument, elle répertoriera le contenu de ce répertoire. Dans l'exemple suivant, le répertoire /etc/ppp est utilisé comme argument ; le résultat résultant est une liste de fichiers contenus dans le répertoire /etc/ppp :

```
sysadmin@localhost:~$ ls /etc/ppp
ip-down.d  ip-up.d
```

La commande ls accepte également plusieurs arguments. Pour lister le contenu des répertoires /etc/ppp et /etc/ssh, transmettez-les tous les deux en arguments :

```
sysadmin@localhost:~$ ls /etc/ppp /etc/ssh
/etc/ppp:
ip-down.d  ip-up.d

/etc/ssh:
moduli          ssh_host_ecdsa_key      ssh_host_rsa_key
ssh_config      ssh_host_ecdsa_key.pub  ssh_host_rsa_key
ssh_host_dsa_key ssh_host_ed25519_key    ssh_import_id
ssh_host_dsa_key.pub ssh_host_ed25519_key.pub sshd_config
```

Certaines commandes, comme la commande cp (copier le fichier) et la commande mv (déplacer le fichier), nécessitent toujours au moins deux arguments : un fichier source et un fichier de destination.

```
cp SOURCE... DESTINATION
```

Dans l'exemple ci-dessous, nous allons copier la clé publique ssh rsa appelée `ssh_host_rsa_key.pub` qui réside dans le répertoire `/etc/ssh`, dans le répertoire `/home/sysadmin/Documents` et vérifier qu'elle s'y trouve :

```
sysadmin@localhost:~$ cp /etc/ssh/ssh_host_rsa_key.pub /home/sysadmin/Documents
sysadmin@localhost:~$ ls ~/Documents
School      alpha.txt  linux.txt  profile.txt
Work        animals.txt longfile.txt red.txt
adjectives.txt food.txt   newhome.txt spelling.txt
alpha-first.txt hello.sh   numbers.txt ssh_host_rsa_key.pub
alpha-second.txt hidden.txt os.csv      words
alpha-third.txt letters.txt people.csv
```

2.3.1.1 Citation

Les arguments qui contiennent des caractères inhabituels comme des espaces ou des caractères non alphanumériques devront généralement être cités, soit en les plaçant entre guillemets doubles ou simples. Les guillemets doubles empêcheront le shell d'interpréter certains de ces caractères spéciaux ; Les guillemets simples empêchent le shell d'interpréter les caractères spéciaux.

Dans la plupart des cas, les guillemets simples sont considérés comme plus sûrs et devraient probablement être utilisés chaque fois que vous avez un argument contenant des caractères qui ne sont pas alphanumériques.

Pour comprendre l'importance des guillemets, considérez la commande **echo**. La commande `echo` affiche le texte sur le terminal et est largement utilisée dans les scripts shell.

```
echo [STRING]...
```

Considérez le scénario suivant dans lequel vous souhaitez répertorier le contenu du répertoire actuel à l'aide de la commande `ls`, puis utiliser la commande `echo` pour afficher la chaîne **hello world!!** sur l'écran.

Vous pouvez d'abord essayer la commande `echo` sans guillemets, malheureusement sans succès :


```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Template
sysadmin@localhost:~$ echo hello world!!
echo hello worldls
hello worldls
```

L'utilisation de guillemets a échoué car le shell interprète les caractères **!!** en tant que caractères shell spéciaux ; dans ce cas, ils signifient "remplacer le **!!** par la dernière commande exécutée". Dans ce cas, la dernière commande était la commande **ls**, donc **ls** a été remplacée par **!!**, puis la commande **echo** a affiché **Hello worldls** à l'écran.

Vous voudrez peut-être essayer les guillemets doubles **"** pour voir s'ils bloquent l'interprétation (ou l'expansion) des caractères d'exclamation **!!**. Les guillemets doubles bloquent l'expansion de certains caractères spéciaux, mais pas de tous.

```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Template
sysadmin@localhost:~$ echo "hello world!!"
echo "hello worldls"
hello worldls
```

L'utilisation de guillemets doubles préserve la valeur littérale de tous les caractères qu'ils entourent, à l'exception des métacaractères tels que le caractère **\$** dollar, le caractère **`** backquote, le caractère **** backslash et le **!** caractère de point d'exclamation. Ces caractères, appelés **jokers**, sont des caractères symboliques qui ont une signification particulière pour le shell. Les caractères génériques sont utilisés pour la globalisation et sont interprétés par le shell lui-même avant de tenter d'exécuter une commande. Les caractères **Glob** sont utiles car ils vous permettent de spécifier des modèles qui facilitent la correspondance des noms de fichiers dans la ligne de commande.

Note

Le **globbing** sera abordé plus en détail plus tard dans le cours.

Si vous placez du texte entre guillemets simples **'**, alors tous les caractères ont leur signification littérale :

```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Template
sysadmin@localhost:~$ echo 'hello world!!'
hello world!!
```

2.3.2 Options

```
command [options] [arguments]
```

Les options peuvent être utilisées avec des commandes pour développer ou modifier le comportement d'une commande. S'il est nécessaire d'ajouter des options, elles peuvent être spécifiées après le nom de la commande. Les options courtes sont spécifiées par un trait d'union suivi d'un seul caractère. Les options courtes sont la façon dont les options étaient traditionnellement spécifiées.

Dans l'exemple suivant, l'option `-l` est fournie à la commande `ls`, ce qui entraîne un affichage long :

```
sysadmin@localhost:~$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Desktop
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Documents
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Music
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Pictures
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Public
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Templates
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Videos
```

Souvent, le caractère est choisi pour être mnémotechnique en fonction de son objectif, comme choisir la lettre `l` pour long ou `r` pour inverse. Par défaut, la commande `ls` imprime les résultats par ordre alphabétique, donc l'ajout de l'option `-r` imprime les résultats dans l'ordre alphabétique inverse.

```
sysadmin@localhost:~$ ls -r
Videos Templates Public Pictures Music Downloads Documents
```

Dans la plupart des cas, les options peuvent être utilisées conjointement avec d'autres options. Ils peuvent être donnés sous forme d'options distinctes comme `-l -r` ou combinées comme `-lr`. La combinaison de ces deux options entraînerait une longue liste dans l'ordre alphabétique inverse :

```
sysadmin@localhost:~$ ls -l -r
total 0
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Videos
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Templates
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Public
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Pictures
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Music
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Documents
drwxr-xr-x 1 sysadmin sysadmin 0 Sep 18 22:25 Desktop
```

Plusieurs options simples peuvent être données sous forme d'options distinctes comme `-a -l -r` ou combinées comme `-alr`. Le résultat de tous ces exemples serait le même :

```
ls -l -a -r
ls -rla
ls -a -lr
```

Généralement, les options courtes peuvent être combinées avec d'autres options courtes dans n'importe quel ordre. L'exception à cela est lorsqu'une option nécessite un argument.

Par exemple, l'option `-w` de la commande `ls` spécifie la largeur de la sortie souhaitée et nécessite donc un argument. Si elle est combinée avec d'autres options, l'option `-w` peut être spécifiée en dernier, suivie de son argument et rester valide, comme dans `ls -lr w 40`, qui spécifie une largeur de sortie de 40 caractères. Sinon, l'option `-w` ne peut pas être combinée avec d'autres options et doit être donnée séparément.

```
sysadmin@localhost:~$ ls -lr -w 40
total 32
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Videos
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Templates
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Public
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Pictures
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Music
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Downloads
drwxr-xr-x 4 sysadmin sysadmin 4096 Feb 22 16:32 Documents
drwxr-xr-x 2 sysadmin sysadmin 4096 Feb 22 16:32 Desktop
```

Si vous utilisez plusieurs options nécessitant des arguments, ne les combinez pas. Par exemple, l'option `-T`, qui spécifie la taille des tabulations, nécessite également un argument. Afin de prendre en compte les deux arguments, chaque option est donnée séparément :

```
sysadmin@localhost:~$ ls -w 40 -T 12
Desktop Music  Templates
Documents Pictures Videos
Downloads Public
```

Certaines commandes prennent en charge des options supplémentaires dépassant un seul caractère. Les options longues pour les commandes sont précédées d'un double trait d'union - et la signification de l'option est généralement le nom de l'option, comme l'option --all, qui répertorie tous les fichiers, y compris ceux masqués. Par exemple:

```
sysadmin@localhost:~$ ls --all
.          .bashrc    .selected_editor Downloads Public
..         .cache Desktop      Music  Templates
.bash_logout .profile Documents Pictures Videos
```

Pour les commandes prenant en charge les options longues et courtes, exécutez la commande en utilisant simultanément les options longue et courte :

```
sysadmin@localhost:~$ ls --all --reverse -t
.profile      Templates Music      Desktop  ..
.bash_logout Public Downloads .selected_editor .cache
Videos        Pictures Documents .bashrc  .
```

Les commandes qui prennent en charge les options longues prennent souvent également en charge les arguments qui peuvent être spécifiés avec ou sans symbole égal (le résultat des deux commandes est le même) :

```
ls --sort time
ls --sort=time
```

```
sysadmin@localhost:~$ ls --sort=time
Desktop Documents Downloads Music Pictures Public Template
```

2.4 Scénarios

Une exception à la syntaxe de commande de base utilisée est la commande **exec**, qui prend une autre commande à exécuter comme argument. La particularité des commandes exécutées avec **exec** est qu'elles remplacent le shell en cours d'exécution.

Une utilisation courante de la commande **exec** est ce que l'on appelle les **scripts wrapper**. Si le but d'un script est simplement de configurer et de lancer un autre programme, on parle alors de script wrapper.

Un script wrapper utilise souvent ce qui suit comme dernière ligne du script pour exécuter un autre programme.

```
exec program
```

Un script écrit de cette manière évite qu'un shell continue de s'exécuter pendant que le programme qu'il a lancé est en cours d'exécution, le résultat est que cette technique économise des ressources (comme la RAM).

Bien que la redirection des entrées et des sorties vers un script soit abordée dans une autre section, il convient également de mentionner que la commande **exec** peut être utilisée pour provoquer la redirection d'une ou plusieurs instructions dans un script.

2.5 Affichage des informations système

La commande **uname** affiche les informations système. Cette commande affichera Linux par défaut lorsqu'elle sera exécutée sans aucune option.

```
sysadmin@localhost:~$ uname  
Linux
```

La commande **uname** est utile pour plusieurs raisons, notamment lorsque vous devez déterminer le nom de l'ordinateur ainsi que la version actuelle du noyau utilisé.

Pour afficher des informations supplémentaires sur le système, vous pouvez utiliser l'une des nombreuses options disponibles pour la commande **uname**. Par exemple, pour afficher toutes les informations sur le système, utilisez l'option **-a** avec la commande **uname** :

```
sysadmin@localhost:~$ uname -a
Linux localhost 4.4.0-72-generic #93~14.04.1-Ubuntu SMP Fri Mar
2017 x86_64 x86_64 x86_64 GNU/Linux
```

Pour afficher des informations sur la version du noyau exécutée par le système, utilisez l'option `-r` :

```
sysadmin@localhost:~$ uname -r
4.4.0-72-generic
```

Les options de la commande `uname` sont résumées ci-dessous :

Short Option	Long Option	Prints
<code>-a</code>	<code>--all</code>	All information
<code>-s</code>	<code>--kernel-name</code>	Kernel name
<code>-n</code>	<code>--node-name</code>	Network node name
<code>-r</code>	<code>--kernel-release</code>	Kernel release
<code>-v</code>	<code>--kernel-version</code>	Kernel version
<code>-m</code>	<code>--machine</code>	Machine hardware name
<code>-p</code>	<code>--processor</code>	Processor type or unknown
<code>-i</code>	<code>--hardware-platform</code>	Hardware platform or unknown
<code>-o</code>	<code>--operating-system</code>	Operating system
	<code>--help</code>	Help information
	<code>--version</code>	Version information

2.6 Répertoire actuel

L'une des commandes les plus simples disponibles est la commande **pwd**, qui est un acronyme pour imprimer le répertoire de travail. Lorsqu'elle est exécutée sans aucune option, la commande `pwd`

va afficher le nom du répertoire où se trouve actuellement l'utilisateur dans le système de fichiers. Lorsqu'un utilisateur se connecte à un système, il est normalement placé dans son répertoire personnel où résident les fichiers qu'il crée et contrôle. Lorsque vous naviguez dans le système de fichiers, il est souvent utile de savoir dans quel répertoire vous vous trouvez.

```
sysadmin@localhost:~$ pwd
/home/sysadmin
```

Notez que nos machines virtuelles utilisent une invite qui affiche le répertoire de travail actuel, souligné par la couleur bleue. Dans la première invite ci-dessus, le caractère **tilde bleu** `~` équivaut à `/home/sysadmin`, représentant le répertoire personnel de l'utilisateur :

```
sysadmin@localhost:~$
```

Après avoir changé de répertoire, le nouvel emplacement peut également être confirmé dans la nouvelle invite à l'aide de la commande `pwd` et est à nouveau affiché en bleu :

```
sysadmin@localhost:~$ cd Documents/
sysadmin@localhost:~/Documents$ pwd
/home/sysadmin/Documents
```

Pour revenir au répertoire personnel après avoir changé d'emplacement, utilisez la commande `cd` change directory sans aucun argument :

```
sysadmin@localhost:~/Documents$ cd
sysadmin@localhost:~$
```

2.7 Informations sur les commandes

La commande **type** affiche des informations sur un type de commande. Par exemple, si vous avez entré le **type** `ls` à l'invite de commande, il sera renvoyé que la commande `ls` est en fait un alias pour la commande `ls --color=auto` :

```
sysadmin@localhost:~$ type ls
ls is aliased to `ls --color=auto'
```

L'utilisation de l'option **-a** avec la commande `type` renverra tous les emplacements des fichiers contenant une commande ; également appelé fichier exécutable :

```
sysadmin@localhost:~$ type -a ls
ls is aliased to `ls --color=auto'
ls is /bin/ls
```

Dans la sortie ci-dessus, le chemin du fichier `/bin/ls` est l'emplacement du fichier de la commande `ls`.

Cette commande est utile pour obtenir des informations sur les commandes et leur emplacement sur le système. Pour les commandes internes, comme la commande `pwd`, la commande `type` les identifiera comme des commandes intégrées au shell :

```
sysadmin@localhost:~$ type pwd
pwd is a shell builtin
```

Pour les commandes externes comme la commande `ip`, la commande `type` renverra l'emplacement de la commande, dans ce cas, le répertoire `/sbin` :

```
sysadmin@localhost:~$ type ip
ip is /sbin/ip
```

Considère ceci

Le répertoire `/bin` contient les programmes exécutables nécessaires au démarrage d'un système, les commandes couramment utilisées et d'autres programmes nécessaires aux fonctionnalités de base du système.

Le répertoire `/sbin` contient également des programmes exécutables ; principalement des commandes et des outils conçus pour l'administration du système.

Si une commande ne se comporte pas comme prévu ou si une commande n'est pas accessible, il peut être utile de savoir où le shell trouve la commande.

La commande **which** recherche l'emplacement d'une commande dans le système en recherchant la variable `PATH`.

```
sysadmin@localhost:~$ which ls
/bin/ls
```


Note

La variable PATH contient une liste de répertoires utilisés pour rechercher les commandes saisies par l'utilisateur.

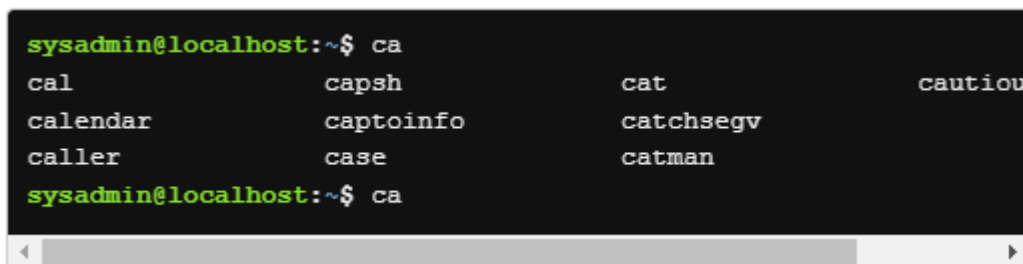
La variable PATH sera abordée plus en détail plus tard dans le cours.

2.8 Achèvement des commandes

Un outil utile du shell Bash est la possibilité de compléter automatiquement les commandes et leurs arguments. Comme de nombreux shells de ligne de commande, Bash propose la complétion de ligne de commande, dans laquelle vous tapez quelques caractères d'une commande (ou son argument de nom de fichier), puis appuyez sur la touche Tab. Le shell Bash complétera automatiquement la commande (ou son argument de nom de fichier). Par exemple, si vous tapez ech et appuyez sur Tab, le shell terminera automatiquement la commande echo pour vous.

Il y aura des moments où vous tapez un caractère ou deux et appuyez sur la touche Tab, pour découvrir que Bash ne termine pas automatiquement la commande. Cela se produit lorsque vous n'avez pas tapé suffisamment de caractères pour correspondre à une seule commande. Cependant, appuyer une deuxième fois sur la touche Tab dans cette situation affichera les complétions possibles (commandes possibles) disponibles.

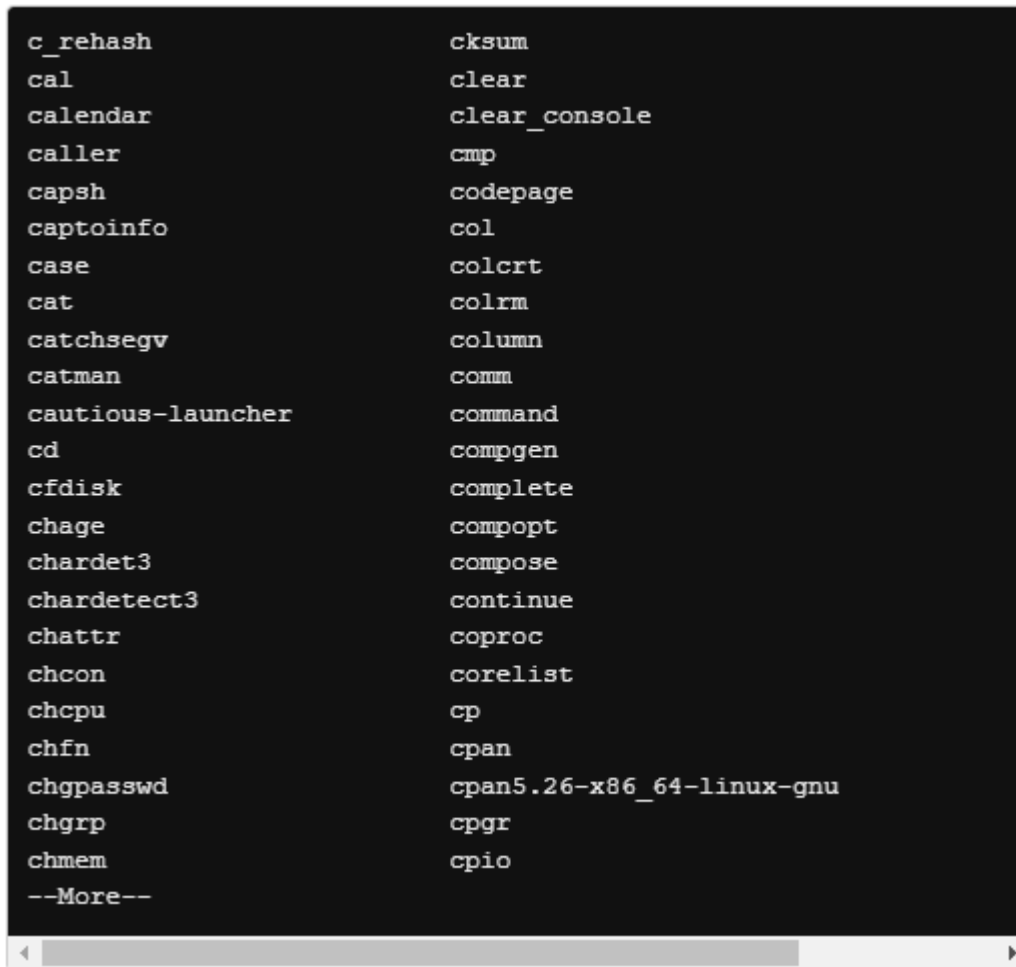
Un bon exemple serait si vous tapiez **ca** et appuyiez sur Tab ; alors rien ne serait affiché. Si vous appuyez une deuxième fois sur Tab, les méthodes possibles pour exécuter une commande commençant par ca s'afficheront :



```
sysadmin@localhost:~$ ca
cal          capsh          cat          cautiou
calendar    captainfo       catchsegv
caller      case           catman
sysadmin@localhost:~$ ca
```

Une autre possibilité peut se produire lorsque vous avez tapé trop peu de données pour correspondre de manière unique à un seul nom de commande. S'il y a plus de correspondances possibles avec ce que vous avez tapé que ce qui peut être facilement affiché, le système utilisera un téléavertisseur pour afficher le résultat, ce qui vous permettra de faire défiler les correspondances possibles.

Par exemple, si vous tapez simplement **c** et appuyez deux fois sur la touche Tab, le système vous proposera de nombreuses correspondances que vous pourrez parcourir :



```
c_rehash      cksum
cal           clear
calendar      clear_console
caller        cmp
capsh         codepage
captinfo      col
case          colcrt
cat           colrm
catchsegv     column
catman        comm
cautious-launcher command
cd            compgen
cfdisk        complete
chage         compopt
chardet3      compose
chardetect3   continue
chattr        coproc
chcon         corelist
chcpu         cp
chfn          cpan
chgpasswd     cpan5.26-x86_64-linux-gnu
chgrp         cpgr
chmem         cpio
--More--
```

Vous ne souhaitez peut-être pas faire défiler toutes ces options ; dans ce cas, appuyez sur Q pour quitter le téléavertisseur. Dans une situation comme celle-ci, vous devriez probablement continuer à taper plus de caractères pour obtenir une correspondance plus raffinée.

Une erreur courante lors de la saisie de commandes est de mal orthographier le nom de la commande. Non seulement vous taperez les commandes plus rapidement, mais vous taperez avec plus de précision si vous utilisez la complétion des commandes. L'utilisation de la touche Tab pour terminer automatiquement la commande permet de garantir que la commande est correctement saisie.

Notez que la complétion fonctionne également pour les arguments des commandes lorsque les arguments sont des noms de fichiers ou de répertoires.

2.9 Obtenir de l'aide

Comme mentionné précédemment, UNIX était le système d'exploitation à partir duquel les fondations de Linux ont été construites. Les développeurs d'UNIX ont créé des documents d'aide appelés **pages de manuel**.

Se référer à la page de manuel d'une commande vous fournira l'idée de base derrière l'objectif de la commande, ainsi que des détails concernant les options de la commande et une description de ses fonctionnalités.

2.9.1 Affichage des pages de manuel

Pour afficher une page de manuel pour une commande, exécutez la commande **man** dans une fenêtre de terminal.

```
man command
```

Par exemple, le graphique suivant montre la page de manuel partielle de la commande **cal** :

```
sysadmin@localhost:~$ man cal
```

```
CAL(1)                                BSD General Commands Manual

NAME
    cal, ncal -- displays a calendar and the date of Easter

SYNOPSIS
    cal [-3h jy] [-A number] [-B number] [[month] year]
    cal [-3hj] [-A number] [-B number] -m month [year]
    ncal [-3bhjJpwySM] [-A number] [-B number] [-s country_code]
        year]
    ncal [-3bhJeoSM] [-A number] [-B number] [year]
    ncal [-CN] [-H yyyy-mm-dd] [-d yyyy-mm]

DESCRIPTION
    The cal utility displays a simple calendar in traditional format. It
    offers an alternative layout, more options and the date of Easter.
    The new format is a little cramped but it makes a year fit on a
    single page. If arguments are not specified, the current month is
    displayed.

    The options are as follows:

    -h          Turns off highlighting of today.
```

Considère ceci

Il est possible d'imprimer des pages de manuel depuis la ligne de commande. Si vous souhaitez envoyer une page de manuel à une imprimante par défaut à partir d'une machine locale, vous devez exécuter la commande suivante :

```
man -t command | lp
```

2.9.2 Contrôle de l'affichage de la page de manuel

La commande `man` utilise un pager pour afficher les documents. Généralement, ce pager est la commande **less**, mais sur certaines distributions, il peut s'agir de la commande **more**. Les deux sont très similaires dans leur fonctionnement et seront discutés plus en détail dans un chapitre ultérieur.

Pour afficher les différentes commandes de mouvement disponibles, utilisez la touche **H** ou **Maj+H** lors de l'affichage d'une page de manuel. Cela affichera une page d'aide.

Note

Si vous travaillez sur une distribution Linux qui utilise la commande **more** comme pager, votre sortie sera différente de l'exemple partiel présenté ici.

Les téléavertisseurs seront abordés plus en détail plus tard dans le cours.

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.
A key preceded by a caret indicates the Ctrl key; thus ^K

h H Display this help.
q :q Q :Q ZZ Exit.

MOVING

e	^E	j	^N	CR	*	Forward one line (or N lines).
y	^Y	k	^K	^P	*	Backward one line (or N lines).
f	^F	^V	SPACE		*	Forward one window (or N lines).
b	^B	ESC-v			*	Backward one window (or N lines).
z					*	Forward one window (and set window to N)
w					*	Backward one window (and set window to N)
	ESC-SPACE				*	Forward one window, but don't stop at en
d	^D				*	Forward one half-window (and set half-wi
u	^U				*	Backward one half-window (and set half-wi
	ESC-)	RightArrow			*	Left one half screen width (or N positio
	ESC-(LeftArrow			*	Right one half screen width (or N positio

Pour quitter le RÉSUMÉ DE MOINS DE COMMANDES, tapez Q.

Si votre distribution utilise la commande **less**, vous pourriez être un peu dépassé par le grand nombre de « commandes » disponibles. Le tableau suivant fournit un résumé des commandes les plus utiles :

Command	Function
Return (or Enter)	Go down one line
Space	Go down one page
<code>/term</code>	Search for <code>term</code>
<code>n</code>	Find next search item
<code>1G</code>	Go to the beginning of the page
<code>G</code>	Go to the end of the page
<code>h</code>	Display help
<code>q</code>	Quit man page

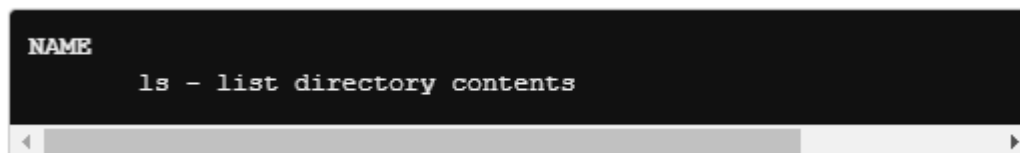
2.9.3 Sections dans les pages de manuel

Chaque page de manuel est divisée en sections. Chaque section est conçue pour fournir des informations spécifiques sur une commande. Bien qu'il existe des sections communes que vous verrez dans la plupart des pages de manuel, certains développeurs créent également des sections que vous ne verrez que dans une page de manuel spécifique.

Le tableau suivant décrit certaines des sections les plus courantes que vous trouverez dans les pages de manuel :

NOM

Fournit le nom de la commande et une très brève description.



SYNOPSIS

Un bref résumé de l'interface de la commande ou de la fonction. Un résumé de l'apparence de la syntaxe de ligne de commande du programme.

SYNOPSIS

```
ls [OPTION]... [FILE]...
```

DESCRIPTION

Fournit une description plus détaillée de la commande.

DESCRIPTION

```
List information about the FILEs (the current director  
Sort entries alphabetically if none of -cftuvSUX nor --so  
fied.
```

OPTIONS

Répertorie les options de la commande ainsi qu'une description de la façon dont elles sont utilisées. Souvent, ces informations se trouvent dans la section DESCRIPTION et non dans une section OPTIONS distincte.

```
-a, --all
```

```
do not ignore entries starting with .
```

```
-A, --almost-all
```

```
do not list implied . and ..
```

```
--author
```

```
with -l, print the author of each file
```

```
-b, --escape
```

```
print C-style escapes for nongraphic characters
```

```
--block-size=SIZE
```

```
scale sizes by SIZE before printing them; e.g., '-  
prints sizes in units of 1,048,576 bytes; see SIZE
```

FICHIERS

Répertorie les fichiers associés à la commande ainsi qu'une description de la façon dont ils sont utilisés. Ces fichiers peuvent être utilisés pour configurer les fonctionnalités plus avancées de la commande. Souvent, ces informations se trouvent dans la section DESCRIPTION et non dans une section FICHIERS distincte.

AUTEUR

Fournit le nom de la personne qui a créé la page de manuel et (parfois) comment contacter cette personne.

```
AUTHOR
    Written by Richard M. Stallman and David MacKenzie.
```

RAPPORT DE BOGUES

Fournit des détails sur la façon de signaler des problèmes de la commande.

```
REPORTING BUGS
    GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
    Report ls translation bugs to <http://translationproject.org/>
```

DROITS D'AUTEUR

Fournit des informations de base sur les droits d'auteur.

```
COPYRIGHT
    Copyright (C) 2017 Free Software Foundation, Inc.  License
    GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
    This is free software: you are free to change and re
    There is NO WARRANTY, to the extent permitted by law.
```

VOIR ÉGALEMENT

Vous donne une idée de l'endroit où vous pouvez trouver des informations supplémentaires. Cela inclut souvent d'autres commandes liées à cette commande.

```
SEE ALSO
    Full documentation at: <http://www.gnu.org/software/coreutils/>
    or available locally via: info '(coreutils) ls invocation
```


2.9.4 Synopsis des pages de manuel

La section SYNOPSIS d'une page de manuel peut être difficile à comprendre, mais c'est une ressource précieuse car elle fournit un exemple concis de la façon d'utiliser la commande. Par exemple, considérons un exemple de SYNOPSIS pour la commande **cal** :

SYNOPSIS

```
cal [-3hjy] [-A number] [-B number] [[month] year]
```

Les crochets [] sont utilisés pour indiquer que cette fonctionnalité n'est pas requise pour exécuter la commande. Par exemple, [-3hjy] signifie que vous pouvez utiliser les options -3, -h, -j, -y, mais aucune de ces options n'est requise pour que la commande cal fonctionne correctement.

Le deuxième jeu de crochets [-A number] permet de préciser le nombre de mois à ajouter à la fin de l'affichage.

Le troisième jeu de crochets [numéro -B] vous permet de spécifier le nombre de mois à ajouter au début de l'affichage.

La quatrième série de crochets [[mois] année] démontre une autre caractéristique ; cela signifie que vous pouvez spécifier une année seule, mais si vous spécifiez un mois, vous devez également spécifier une année.

Un autre composant du SYNOPSIS qui pourrait prêter à confusion peut être vu dans le SYNOPSIS de la commande date :

SYNOPSIS

```
date [OPTION]... [+FORMAT]
```

```
date [-u|--utc|--universal] [MMDDhhmm [CC]YY] [.ss]]
```

Dans ce SYNOPSIS, il existe deux syntaxes pour la commande date. Le premier sert à afficher la date sur le système tandis que le second sert à régler la date.

Les points de suspension... après [OPTION] indiquent que [OPTION] peut être répétée.

De plus, la notation [-u|--utc|--universal] signifie que vous pouvez utiliser soit l'option -u, soit l'option --utc, soit l'option --universal. Généralement, cela signifie que les trois options font réellement la même chose, mais parfois ce format (utilisation du caractère |) est utilisé pour indiquer que les options ne peuvent pas être utilisées en combinaison, comme un **ou** logique.

2.9.5 Recherche dans une page de manuel

Pour rechercher un terme dans une page de manuel, tapez le caractère / suivi du terme et appuyez sur la touche Entrée. Le programme effectuera une recherche depuis l'emplacement actuel vers le bas de la page pour essayer de localiser et de mettre en évidence le terme.

Si le terme n'est pas trouvé ou si vous avez atteint la fin des correspondances, le programme signalera **Modèle non trouvé** (appuyez sur Retour). Si une correspondance est trouvée et que vous souhaitez passer à la correspondance suivante du terme, appuyez sur **N**. Pour revenir à une correspondance précédente du terme, appuyez sur **Maj+N**.

2.9.6 Sections de la page de manuel

Jusqu'à présent, nous affichions des pages de manuel pour les commandes. Cependant, les fichiers de configuration comportent parfois également des pages de manuel. Les fichiers de configuration (parfois appelés fichiers système) contiennent des informations utilisées pour stocker des informations sur le système d'exploitation ou les services.

De plus, il existe plusieurs types de commandes (commandes utilisateur, commandes système et commandes d'administration), ainsi que d'autres fonctionnalités qui nécessitent une documentation, telles que les bibliothèques et les composants du noyau.

En conséquence, il existe des milliers de pages de manuel sur une distribution Linux typique. Pour organiser toutes ces pages de manuel, les pages sont classées par sections, tout comme chaque page de manuel est divisée en sections.

Par défaut, il y a neuf sections de pages de manuel :

- Programmes exécutables ou commandes shell
- Appels système (fonctions fournies par le noyau)
- Appels de bibliothèque (fonctions au sein des bibliothèques de programmes)
- Fichiers spéciaux (généralement trouvés dans /dev)
- Formats de fichiers et conventions, par ex. /etc/mot de passe
- Jeux

- Divers (y compris les packages de macros et les conventions)
- Commandes d'administration système (généralement uniquement pour root)
- Routines du noyau [non standard]

Lorsque vous utilisez la commande `man`, elle recherche chacune de ces sections dans l'ordre jusqu'à ce qu'elle trouve la première correspondance. Par exemple, si vous exécutez la commande `man cal`, la première section (Programmes exécutables ou commandes shell) est recherchée pour une page de manuel appelée `cal`. S'il n'est pas trouvé, la deuxième section est recherchée. Si aucune page de manuel n'est trouvée après avoir recherché toutes les sections, vous recevrez un message d'erreur :

```
sysadmin@localhost:~$ man zed
No manual entry for zed
```

Considère ceci

En règle générale, il y a beaucoup plus de sections que les neuf standards. Les concepteurs de logiciels personnalisés créent des pages de manuel en utilisant des noms de section non standard, par exemple « 3p ».

2.9.6.1 Déterminer quelle section

Pour déterminer à quelle section appartient une page de manuel spécifique, examinez la valeur numérique sur la première ligne de la sortie de la page de manuel. Par exemple, si vous exécutez la commande `man cal`, vous verrez que la commande `cal` appartient à la première section des pages de manuel :

```
sysadmin@localhost:~$ man cal
```

```
CAL(1) BSD General Commands Manual
```

2.9.6.2 Spécification d'une section

Dans certains cas, vous devrez spécifier la section afin d'afficher la page de manuel correcte. Ceci est nécessaire car il y aura parfois des pages de manuel portant le même nom dans différentes sections.

Par exemple, il existe une commande appelée **passwd** qui vous permet de modifier votre mot de passe. Il existe également un fichier appelé **passwd** qui stocke les informations de compte. La commande et le fichier ont une page de manuel.

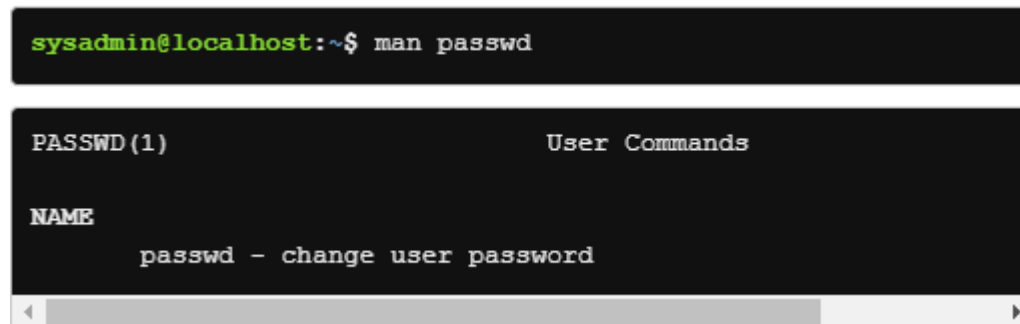
La commande `passwd` est une commande utilisateur, donc la commande suivante affichera par défaut la page de manuel de la commande `passwd`, située dans la première section :

```
sysadmin@localhost:~$ man passwd
```

```
PASSWD(1)                                User Commands
```

```
NAME
```

```
passwd - change user password
```

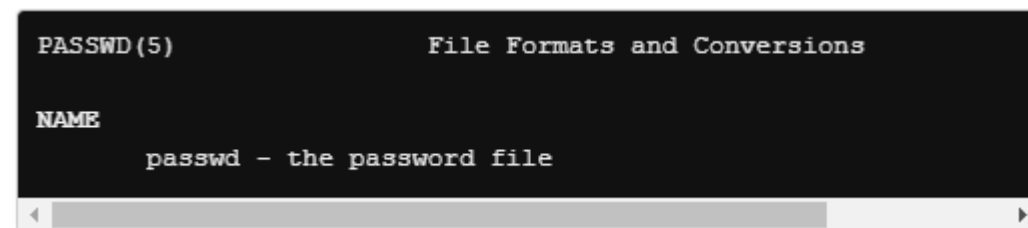
A terminal window showing the output of the 'man passwd' command. It displays the first section of the manual page, titled 'PASSWD(1) User Commands'. Below the title, it shows the 'NAME' section with the description 'passwd - change user password'. A horizontal scrollbar is visible at the bottom of the terminal window.

Pour spécifier une section différente, fournissez le numéro de la section comme premier argument de la commande `man`. Par exemple, la commande `man 5 passwd` recherchera la page de manuel `passwd` dans la section 5 uniquement :

```
PASSWD(5)                                File Formats and Conversions
```

```
NAME
```

```
passwd - the password file
```

A terminal window showing the output of the 'man 5 passwd' command. It displays the fifth section of the manual page, titled 'PASSWD(5) File Formats and Conversions'. Below the title, it shows the 'NAME' section with the description 'passwd - the password file'. A horizontal scrollbar is visible at the bottom of the terminal window.

2.9.6.3 Recherche par nom

Parfois, il n'est pas clair dans quelle section une page de manuel est stockée. Dans des cas comme celui-ci, vous pouvez rechercher une page de manuel par son nom.

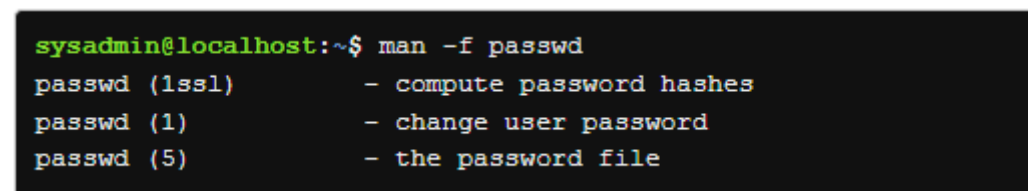
L'option `-f` de la commande `man` affichera les pages de manuel qui correspondent, ou correspondent partiellement, à un nom spécifique et fournira une brève description de chaque page de manuel :

```
sysadmin@localhost:~$ man -f passwd
```

```
passwd (1ss1)      - compute password hashes
```

```
passwd (1)         - change user password
```

```
passwd (5)         - the password file
```

A terminal window showing the output of the 'man -f passwd' command. It lists three manual pages: 'passwd (1ss1) - compute password hashes', 'passwd (1) - change user password', and 'passwd (5) - the password file'.

Notez que sur la plupart des distributions Linux, la commande `whatis` fait la même chose que `man -f`. Sur ces distributions, les deux produiront le même résultat.

```
sysadmin@localhost:~$ whatis passwd
passwd (1ss1)      - compute password hashes
passwd (1)         - change user password
passwd (5)         - the password file
```

2.9.6.4 Recherche par mot-clé

Malheureusement, vous ne vous souviendrez pas toujours du nom exact de la page de manuel que vous souhaitez afficher. Dans ces cas, vous pouvez rechercher des pages de manuel correspondant à un mot-clé en utilisant l'option **-k** de la commande **man**.

Par exemple, que se passerait-il si vous saviez que vous souhaitiez une page de manuel indiquant comment modifier votre mot de passe, mais que vous ne vous souveniez pas du nom exact ? Vous pouvez exécuter la commande **man -k password** :

```
sysadmin@localhost:~$ man -k password
chage (1)          - change user password expiry information
chgrp (8)           - update group passwords in batch mode
chpasswd (8)        - update passwords in batch mode
cpgr (8)           - copy with locking the given file to the p
cppw (8)           - copy with locking the given file to the p
```

Avertissement

Lorsque vous utilisez cette option, vous risquez de vous retrouver avec une grande quantité de sortie. La commande précédente, par exemple, a fourni plus de 60 résultats.

N'oubliez pas qu'il existe des milliers de pages de manuel. Par conséquent, lorsque vous recherchez un mot-clé, soyez aussi précis que possible. L'utilisation d'un mot générique, tel que « le », peut donner lieu à des centaines, voire des milliers de résultats.

Notez que sur la plupart des distributions Linux, la commande **apropos** fait la même chose que **man -k**. Sur ces distributions, les deux produiront le même résultat.

Considère ceci

Vous souhaitez en savoir plus sur les pages de manuel ? Exécutez la commande suivante :

```
sysadmin@localhost:~$ man man
```