Chapitre 3
Le shell et les commandes GNU

1. Le shell bash

1.1 Rôle du shell

Même si toutes les distributions proposent des interfaces utilisateur graphiques, un informaticien professionnel travaillant sur un système Linux doit connaître le fonctionnement de l'interpréteur de commandes (shell) et des principales commandes en mode caractère. D'une part, les systèmes serveurs sont généralement installés sans interface graphique, d'autre part il est indispensable de pouvoir gérer les scripts d'exploitation et d'administration écrits en langage shell et combinant des commandes en mode caractère.

L'interpréteur de commandes permet d'exécuter des instructions saisies au clavier ou lues dans un fichier script. Cet interpréteur est le plus souvent un programme de type shell. Le terme shell (coquille), d'origine Unix, est employé en référence au terme **kernel** (noyau) : le shell est une interface « autour » du noyau Linux, fonctionnant en mode caractère.

Il existe plusieurs programmes de type shell, chacun disposant de spécificités propres. Le **Bourne Shell**, du nom de son créateur **Steve Bourne**, est le shell le plus ancien, écrit pour Unix. Le shell a ensuite été normalisé dans le cadre des normes POSIX.

Le shell de référence de la plupart des distributions Linux est le bash (*Bourne Again Shell*), mais il en existe de nombreux autres, dont :

- `sh` : Bourne Shell
- `ksh` : Korn Shell
- `csh` : C Shell
- `zsh` : Z Shell
- `ash` : A Shell
- `dash` : Debian Almquist Shell.

Remarque

Le fichier `/etc/shells` fournit la liste des shells installés sur le système.

1.2 Bash : le shell Linux par défaut

Le shell bash est un dérivé du Bourne Shell. Il est conforme aux normes POSIX mais il ajoute de nombreuses extensions qui lui sont spécifiques.

Remarque

Dans les distributions Debian récentes, le shell par défaut est le dash, une variante très proche du shell bash.

1.2.1 Un shell puissant et libre

Le bash, sous licence open source GNU, est fourni par défaut avec toutes les distributions Linux. Il existe même en version macOS et Windows (via la fonctionnalité Sous-système Windows pour Linux).

Le shell fonctionne en mode ligne de commande. Quand il est lancé depuis un terminal, il s'initialise à partir de différents fichiers, affiche un message de prompt au début d'une ligne d'invite de commande et se place en mode lecture du clavier. Quand la ligne de commande est saisie et validée par la touche [Entrée], le shell interprète son contenu et l'exécute. Une fois l'exécution terminée, le shell affiche à nouveau le prompt et se remet en attente d'une nouvelle ligne.

La séquence de touches [Ctrl] D termine l'exécution du shell. La commande `exit` provoque également la terminaison du shell.

Remarque

Linux, comme Unix, distingue les minuscules des majuscules, dans les commandes, leurs options et arguments, ainsi que dans les noms de fichiers et de répertoires.

1.2.2 L'invite de commandes

Le shell attend des saisies au clavier sur une ligne appelée l'invite de commandes. La chaîne de caractères affichée au début de cette ligne s'appelle le **prompt**.

Le prompt est configurable (par la variable d'environnement `PS1`), son contenu par défaut est variable suivant les distributions. Il affiche en général le nom du compte utilisateur, le répertoire courant et un caractère `$` (compte non-administrateur) ou # (compte administrateur).

Exemple

Prompt par défaut de l'utilisateur `pba` sur le système `srvrh` (distribution RHEL 9) :

```
[pba@srvrh ~]$
```

Prompt par défaut de l'utilisateur `root` (administrateur) sur le système `srvdeb` (distribution Debian 12) :

```
root@srvdeb:~#
```

1.3 Utiliser le shell

1.3.1 La saisie sur la ligne de commande

Sur la ligne de commande, on peut déplacer le curseur avec les touches [Flèche à droite] et [Flèche à gauche], et effacer des caractères avec les touches [Retour arrière] ou [Suppr]. pour déclencher l'exécution, il faut appuyer sur la touche [Entrée].

Les raccourcis-clavier suivants peuvent être utilisés :

- **[Ctrl] A** : aller au début de la ligne.
- **[Ctrl] E** : aller en fin de ligne.
- **[Ctrl] L** : effacer le contenu de l'écran, et afficher l'invite en haut de celui-ci.
- **[Ctrl] U** : effacer la ligne jusqu'au début.
- **[Ctrl] K** : effacer la ligne jusqu'à la fin.

Exemples

Commande d'affichage de la date.

```
$ date
lun. 15 mai 2023 09:40:20 CEST
```

Commande d'affichage du chemin d'accès du répertoire courant.

```
$ pwd
/home/pba
```

1.3.2 Syntaxe générale des commandes

La plupart des commandes fournies avec les distributions Linux sont d'origine Unix, mais ont été réécrites dans le cadre du projet open source GNU. Leur syntaxe peut varier d'une version à l'autre.

Les commandes GNU/Linux ont en général la syntaxe suivante :

```
Commande [options] [arguments]
```

Une commande peut n'avoir ni option, ni argument. Les options sont le plus souvent identifiées par un caractère précédé d'un tiret : `-l`, `-p`, `-s`, etc. Si la commande accepte plusieurs options, on peut les spécifier les unes après les autres en les séparant par des espaces : `-l -r -t`, ou les grouper derrière un seul tiret : `-lrt`. L'ordre des options n'a pas d'importance, les deux syntaxes précédentes produisent le même résultat.

Remarque

Certaines options attendent un argument, par exemple un nom de fichier. Dans ce cas, on les séparera des autres : `-lrt -f monfichier` ou on les placera en dernière position : `-lrtf monfichier`.

Les arguments sont des chaînes de caractères séparées par un caractère espace ou tabulation. Si un argument doit contenir un espace, il faut l'encadrer par des guillemets simple `''` ou doubles `""`.

1.3.3 Exemple de commande : cal

La commande `cal` admet plusieurs options et arguments. Appelée seule, elle affiche le calendrier du mois en cours.

Exemple

```
$ cal
      mai 2023
lu ma me je ve sa di
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

La commande admet deux arguments optionnels. Si un seul est précisé, il s'agit de l'année, et l'intégralité du calendrier de cette année est affichée. Si deux arguments sont précisés, le premier est le mois, le second l'année.

Exemple

```
$ cal 12 1975
    décembre 1975
lu ma me je ve sa di
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

La commande prend quelques options, variables selon la version installée.

L'option `-m` (*monday*) affiche les jours de la semaine en commençant par lundi.

L'option `-s` (*sunday*) affiche les jours de la semaine en commençant par dimanche.

Exemple

```
$ cal -s 12 1975
    décembre 1975
di lu ma me je ve sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

L'option `-m3` permet d'afficher le mois précédent et le mois suivant le mois spécifié ou, sans argument, le mois courant.

Exemple

```
$ cal -m3 12 1975
cal -m3 12 1975
    novembre 1975         décembre 1975         janvier 1976
lu ma me je ve sa di  lu ma me je ve sa di  lu ma me je ve sa di
                1  2   1  2  3  4  5  6  7            1  2  3  4
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
24 25 26 27 28 29 30  29 30 31              26 27 28 29 30 31
```

Avec une distribution de type Debian, on peut utiliser la commande similaire `ncal` qui possède une syntaxe un peu différente (la commande `cal` exécutant en fait `ncal`).

Exemple

Pour obtenir le même résultat qu'avec l'exemple précédent :

```
$ ncal -b -M -3 12 1975
   Novembre 1975         Décembre 1975          Janvier 1976
lu ma me je ve sa di  lu ma me je ve sa di  lu ma me je ve sa di
                1  2   1  2  3  4  5  6  7            1  2  3  4
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
24 25 26 27 28 29 30  29 30 31              26 27 28 29 30 31
```

1.3.4 Enchaîner les commandes

On peut indiquer plusieurs commandes sur une même ligne de commande, en les séparant par un point-virgule. Elles seront exécutées successivement.

Exemple

```
$ date; pwd; cal -m
lun. 15 mai 2023 11:30:26 CEST
/home/pba
      mai 2023
lu ma me je ve sa di
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

1.3.5 Afficher du texte

La commande `echo` affiche les arguments qu'on lui passe, séparés par un espace et suivis d'un saut de ligne.

Exemple

```
$ echo Bonjour les amis
Bonjour les amis
```

Les arguments peuvent contenir des caractères spéciaux, issus du langage C, à condition de spécifier l'option `-e`. Les plus utilisés sont les suivants :

Séquence	Action
`\n`	Saut de ligne
`\t`	Tabulation horizontale

Séquence	Action
\c	Pas de saut de ligne après l'afffichage des arguments
\b	Retour d'un caractère en arrière
\\	Afficher l'antislash (barre oblique inverse)
\nnn	Afficher le caractère de code octal nnn

Exemple

```
$ echo -e "Salut.\tJe m'appelle XXX\b\b\bPersonne\n"
Salut. Je m'appelle Personne
```

Remarque

Cette commande est surtout utilisée dans les scripts, pour afficher des commentaires, des instructions utilisateurs, des messages d'erreur, etc.

1.3.6 Commandes internes et externes

Il existe deux types de commandes :

– Les **commandes externes** correspondent à des programmes exécutables stockés dans des fichiers. À l'exécution de la commande, le shell détermine l'emplacement du fichier correspondant et, s'il le trouve, lance son exécution en lui passant options et arguments éventuels.

– Les **commandes internes** sont internes au shell et exécutées directement par celui-ci. Le code exécutable de ces commandes fait partie intégrante de celui du shell (primitive du shell). Les commandes `cd` ou `pwd` en sont deux exemples.

Pour distinguer une commande interne d'une commande externe, on peut utiliser la commande interne `type`.

Exemple

```
$ type date
date est haché (/bin/date)
```

La commande `date` est une commande externe, mais son chemin est mémorisé par le shell (hashage).

```
$ type pwd
pwd est une primitive du shell
```

1.3.7 Séquences de contrôle

`[Ctrl] C` : provoque l'interruption de la commande en cours.

`[Ctrl] D` : en début de ligne, termine la saisie si la commande est en lecture du clavier, ou termine le shell courant s'il est en attente du clavier.

Exemple

Sans argument, la commande `sort` trie les lignes saisies au clavier. Pour arrêter la saisie, il faut taper `[Ctrl] D`, en début de ligne :

```
$ sort
bbbbbbbbbbb
aaaaaaa
zzzzzzz
eeeeeeee
[Ctrl]D
aaaaaaa
bbbbbbbbbbb
eeeeeeee
zzzzzzz
```

1.4 Historique des commandes

Le shell conserve dans un fichier du répertoire de connexion de chaque compte utilisateur un historique des lignes de commande, nommé `.bash_history`. Il est possible d'y naviguer avec les touches [Flèche en haut] et [Flèche en bas] sachant que la flèche du haut permet de remonter dans l'historique. On peut modifier ou non la ligne de commande affichée, et demander son exécution avec la touche [Entrée] (quelle que soit la position du curseur sur la ligne de commande).

La commande `history` affiche les dernières lignes de commande saisies.

Exemple

```
$ history
...
 1000  date
 1001  pwd
 1002  uname -a
 1003  ls
 1004  fc -l -5
 1005  history
```

La commande `fc -l` affiche les quinze dernières lignes de commande, en les numérotant. On peut indiquer le nombre de lignes de commande à afficher en le passant en option.

Chapitre 4
Configuration du réseau

1. Configuration du réseau

Ce chapitre présente la configuration de base du réseau, les options avancées et le dépannage.

1.1 Configuration de base du réseau

L'objectif de cette section est de vous apprendre à :

– configurer une interface réseau connectée à un réseau local, par câble ou sans fil, ou à un réseau étendu ;
– en particulier, configurer des sous-réseaux en IPv4 et IPv6.

1.1.1 Compétences principales

– Configurer et gérer des cartes Ethernet.
– Configuration de base de réseaux sans fil.

1.1.2 Éléments mis en œuvre

– `ip`
– `ifconfig`
– `route`
– `arp`
– `iw`

- `iwconfig`
- `iwlist`

1.2 Configuration avancée du réseau

L'objectif de cette section est de vous apprendre à :

- mettre en œuvre différentes méthodes d'authentification de connexion réseau ;
- configurer un système participant à différents réseaux et résoudre différents problèmes de communication.

1.2.1 Compétences principales

- Gestion des tables de routage.
- Outils de configuration et de gestion des interfaces réseau Ethernet.
- Outils d'analyse de l'état des interfaces réseau.
- Outils de supervision et d'analyse du trafic TCP/IP.

1.2.2 Éléments mis en œuvre

- `ip`
- `ifconfig`
- `route`
- `arp`
- `ss`
- `netstat`
- `lsof`
- `ping, ping6`
- `nc`
- `tcpdump`
- `nmap`

1.3 Dépannage réseau

L'objectif de cette section est de vous apprendre à :

- identifier et résoudre les problèmes réseau courants, nécessitant une bonne connaissance des différents fichiers de configuration et des commandes réseau élémentaires.

1.3.1 Compétences principales

- Fichiers de configuration du contrôle d'accès.
- Outils de configuration et de gestion des interfaces réseau Ethernet.
- Outils d'administration des tables de routage.
- Outils de supervision de l'état du réseau.
- Outils de suivi de la configuration du réseau.
- Méthode pour déterminer les périphériques réseau reconnus par le système et leur utilisation.
- Fichiers de configuration de l'initialisation du système (`systemd` et `init System V`).
- Prise en compte de `NetworkManager` et de son rôle dans la configuration du réseau.

1.3.2 Éléments mis en œuvre

- `ip`
- `ifconfig`
- `route`
- `ss`
- `netstat`
- `/etc/network/`, `/etc/sysconfig/network-scripts/`
- `ping`, `ping6`
- `traceroute`, `traceroute6`
- `mtr`
- `hostname`
- Journaux système comme `/var/log/syslog`, `/var/log/messages` et le journal `systemd`
- `dmesg`
- `/etc/resolv.conf`
- `/etc/hosts`
- `/etc/hostname`, `/etc/HOSTNAME`
- `/etc/hosts.allow`, `/etc/hosts.deny`

2. Configuration de base du réseau

Linux est un système d'exploitation particulièrement orienté réseau. La plupart des protocoles réseau modernes y sont implémentés, et la majorité des serveurs d'applications réseau fonctionnant aujourd'hui utilisent Linux.

Cette partie concerne la configuration réseau de base d'un système Linux, en connexion Ethernet et en connexion sans fil, avec IPv4 et IPv6.

Pour connecter une carte d'interface réseau sur un inter-réseau IP, il faut spécifier au minimum deux ou trois paramètres : une adresse IP, un masque de sous-réseau et une passerelle par défaut (sauf si le réseau est strictement local).

2.1 Adresses IPv4 et IPv6

Il existe deux versions usuelles du protocole IP :

– IPv4, la plus ancienne, avec des adresses sur 32 bits.

– IPv6, avec des adresses sur 128 bits.

Un système Linux peut gérer les deux versions de protocole et avoir une ou plusieurs adresses pour chacun des protocoles.

D'autre part, indépendamment de la version du protocole IP, diverses combinaisons entre adresse IP, interface réseau et système sont possibles :

– Un système peut avoir une interface réseau et une adresse IP.

– Un système peut avoir plusieurs interfaces réseau et plusieurs adresses IP.

– Une interface réseau peut avoir une seule adresse IP.

– Une interface réseau peut avoir plusieurs adresses IP.

– Plusieurs interfaces réseau peuvent servir une même adresse IP.

Remarque

Traditionnellement, on emploie le terme d'adresse IP hôte (host address). Ce terme est trompeur, car une machine peut avoir plusieurs interfaces réseau, sur des réseaux IP différents, et donc avoir des adresses IP hôtes différentes. On devrait plutôt dire adresse IP d'interface réseau, encore que, dans certains cas, plusieurs interfaces réseau peuvent servir la même adresse IP.

2.2 Paramétrage de base d'une connexion IPv4

2.2.1 Réseau/sous-réseau

Le protocole IP (*Internet Protocol*) permet de relier entre eux différents réseaux. Pour cela, il faut disposer d'équipements (matériels ou logiciels) participant à plusieurs réseaux et capables de transférer un datagramme IP d'un réseau à un autre (fonction de routage).

Les réseaux IP sont organisés en trois classes, A, B et C, caractérisées par la longueur de leur identifiant réseau : 1 octet pour la classe A, 2 octets pour la classe B et 3 octets pour la classe C.

Deux réseaux IP peuvent communiquer entre eux s'ils sont reliés par au moins un routeur, en traversant éventuellement une succession de réseaux et de routeurs intermédiaires.

Quand le nombre de réseaux interreliés a commencé à devenir très important, il a fallu étendre la notion de réseau, pour permettre aux organisations de découper leur réseau en ensembles interconnectés : les sous-réseaux. Les règles concernant la communication entre les sous-réseaux sont les suivantes :

– Les sous-réseaux sont transparents pour les autres réseaux, qui n'ont besoin de connaître que l'identifiant réseau et l'identifiant de l'hôte destinataire dans son réseau pour communiquer avec lui, quel que soit son sous-réseau.
– Deux sous-réseaux d'un même réseau ne peuvent communiquer entre eux que s'ils sont reliés par au moins un routeur, en traversant éventuellement une succession de sous-réseaux et de routeurs intermédiaires.

Pour identifier les différents sous-réseaux, on utilise une partie de l'identifiant hôte de l'adresse, en plus de l'identifiant réseau. Par conséquent, il n'est pas possible en lisant une adresse IP de savoir à quel sous-réseau de son réseau elle appartient. Il faut donc configurer une information supplémentaire : le nombre de bits de la partie réseau/sous-réseau de l'adresse. Ce paramètre est appelé **masque de sous-réseau** (*subnet mask*).

Dans les documentations, on peut trouver les termes masque de réseau (*net mask*) ou masque de sous-réseau (*subnet mask*). Ils sont équivalents, le premier est plus ancien et date de l'époque où les sous-réseaux étaient peu utilisés.

Remarque

Plus largement, il est aussi possible d'agréger plusieurs réseaux entre eux, pour organiser une sorte de sur-réseau, découpé logiquement en réseaux. On parle alors d'adresses CIDR (Classless Inter Domain Routing).

2.2.2 Adresse IP

Il s'agit de l'adresse IP classique, sur 32 bits. Elle est découpée logiquement en deux parties : l'adresse réseau/sous-réseau, suivie de l'adresse hôte. L'adresse réseau/sous-réseau identifie le réseau/sous-réseau dans l'inter-réseau auquel il participe, l'adresse hôte identifie de façon unique un participant du réseau/sous-réseau.

La répartition des 32 bits de l'adresse entre la partie réseau/sous-réseau et la partie hôte est variable et elle est définie par le masque de sous-réseau.

Une adresse IPv4 peut être spécifiée en utilisant la syntaxe suivante :

```
w.x.y.z[/NbBitsMasque]
```

Soit quatre valeurs entières en notation décimale séparées par un point (notation décimale pointée), spécifiant l'adresse proprement dite, suivies si nécessaire d'un caractère / et du nombre de bits de la partie réseau/sous-réseau, notation dite CIDR (*Classless Internet Domain Routing*).

2.2.3 Masque de sous-réseau

Le masque de sous-réseau permet de déterminer la partie de l'adresse qui identifie le réseau/sous-réseau auquel appartient l'adresse.

Il peut être noté de deux façons :

– Notation classique "masque de sous-réseau" (*subnet mask*) : tous les bits de l'identifiant réseau/sous-réseau sont à 1, tous ceux de l'identifiant hôte sont à 0.

– Notation CIDR : on indique l'identifiant réseau/sous-réseau, tous les bits de l'identifiant hôte à zéro, suivi d'un / et du nombre de bits de l'identifiant réseau/sous-réseau.

Exemple

Pour un identifiant réseau/sous-réseau en `10.1` :

Identifiant réseau/sous-réseau : `10.1.0.0` et masque de sous-réseau : `255.255.0.0`

Notation CIDR : `10.1.0.0/16`

2.2.4 Passerelle par défaut

Si le réseau/sous-réseau n'est pas relié à d'autres réseaux/sous-réseaux, il n'y a pas de passerelle par défaut. Dans le cas contraire, la passerelle par défaut désigne l'adresse IP par défaut vers laquelle faire transiter les datagrammes IP à destination d'un autre réseau/sous-réseau. Si ce paramètre n'est pas spécifié, une carte d'interface réseau ne pourra communiquer qu'avec les nœuds de son réseau/sous-réseau.

2.3 Paramétrage de base d'une connexion IPv6

2.3.1 Adresse IPv6

La version 6 du protocole IP a pour but d'augmenter les fonctionnalités du protocole et de lever certaines de ses limitations. Le passage à une taille d'adresse de 128 bits permet en particulier de répondre au risque de pénurie d'adresses IP dans le cadre d'Internet.

Représentation d'une adresse IPv6

Une adresse IPv6 a une longueur de 128 bits, soit 16 octets. On la représente généralement sous forme de huit éléments de 2 octets. La valeur de chaque élément est notée en hexadécimal, et chaque élément est séparé par le caractère **:**.

Une simplification d'écriture consiste à remplacer une et une seule suite de champs à zéro dans l'adresse par deux caractères **:** contigus.

Exemple

Observons l'adresse IPv6 d'un serveur Google :

```
host www.google.com
www.google.com has address 216.58.215.36
www.google.com has IPv6 address 2a00:1450:4007:80c::2004
```

L'adresse IPv6 retournée correspond à l'adresse complète suivante :

2a00:1450:4007:080c:0000:0000:0000:2004

Structure d'une adresse IPv6

Une adresse IPv6 est composée de trois parties, dans l'ordre de gauche à droite :

- Un préfixe sur 6 octets.
- Un identifiant de sous-réseau, sur 2 octets.
- Un identifiant d'hôte, sur 8 octets.

Exemple

Observons l'adresse IPv6 d'un serveur Google :

```
host www.google.com
www.google.com has address 216.58.215.36
www.google.com has IPv6 address 2a00:1450:4007:80c::2004
```

L'adresse IPv6 a la structure suivante :

Préfixe : `2a00:1450:4007`

Identifiant de sous-réseau : `80c`

Identifiant hôte : `0000:0000:0000:2004`