Chapitre 3

A. Le shell bash 108
B. La gestion des fichiers 115
C. Rechercher des fichiers avec la commande find 132
D. L'éditeur vi 140
E. Les redirections des entrées/sorties standards 146
F. La redirection 147
G. Les commandes filtres 153
H. Autres commandes utilitaires 170
I. La gestion des processus 173
J. Plus loin avec le bash 182
K. Les variables 184
L. Configuration de bash 192
M. Programmation shell 193
N. Multiplexeurs de terminal 214
O. Validation des acquis : questions/réponses 215
P. Travaux pratiques 229

Prérequis

☒Disposer d’un accès utilisateur sur un système Linux.

☒Disposer d'un terminal (console ou graphique).

Objectifs

À la fin de ce chapitre, vous serez en mesure :

☒De travailler avec la ligne de commande.

☒De manipuler l'historique des commandes.

☒D'effectuer des manipulations de base sur le système de fichiers.

☒De rechercher des fichiers dans l'arborescence.

☒De travailler sur des fichiers texte avec l’éditeur de texte vi.

☒De mettre en place des redirections et tubes.

☒D'effectuer des recherches dans des fichiers texte.

☒D'utiliser les filtres et les utilitaires.

☒De gérer les processus.

☒De modifier le chemin de recherche des exécutables, les variables et l'environnement.

☒De modifier la configuration du bash.

☒De créer des scripts.

☒D'effectuer des tests et évaluer la valeur logique d’une expression.

☒D'utiliser les structures de contrôle.

☒De créer des fonctions.

☒D’utiliser un multiplexeur de terminal.

A. Le shell bash

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.

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

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. C'est le shell de référence pour les examens de la certification LPIC-1.

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

a. 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.

☞*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.*

b. 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:~#
```

3. Utiliser le shell

a. 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
```

b. 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.

☞*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 `""`.

c. 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
```

d. 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
```

e. 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
`\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`