Chapitre 3

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`

Prérequis

Les connaissances nécessaires à la certification LPIC-1 :

☒Notions de base sur les périphériques, partitions et systèmes de fichiers.

☒Édition de fichiers.

☒Commandes de gestion de répertoires et de fichiers.

Objectifs

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

☒Déterminer les caractéristiques des différents types de systèmes de fichiers Linux.

☒Créer et gérer les systèmes de fichiers Linux.

☒Configurer et gérer l'espace de swap.

☒Administrer le système de fichiers global Linux et superviser les périphériques SMART.

☒Configurer et administrer l'automontage de systèmes de fichiers de périphériques ou via le réseau.

☒Créer des systèmes de fichiers pour les CD-ROM et autres périphériques.

☒Connaître les caractéristiques des systèmes de fichiers chiffrés.

A. Système de fichiers et périphériques

Ce sujet est divisé en trois parties de poids différents.

1. Administration du système de fichiers Linux

Poids	4
Objectifs	Configurer et gérer le système de fichiers standard Linux, en combinant le montage de plusieurs systèmes de fichiers de types différents.

a. Compétences principales

- Configuration par le fichier `fstab`.
- Outils de gestion de l'espace de swap.
- Identification et montage des systèmes de fichiers par leur UUID.
- Compréhension des unités de montage de `systemd`.

b. Éléments mis en œuvre

- `/etc/fstab`
- `/etc/mtab`
- `/proc/mounts`
- `mount`
- `umount`
- `blkid`
- `sync`
- `swapon`
- `swapoff`

2. Maintenance du système de fichiers Linux

Poids	3
Objectifs	Administrer le système de fichiers Linux avec les outils système pour gérer les types de systèmes de fichiers standards et superviser les périphériques SMART.

a. Compétences principales

- Outils de gestion des systèmes de fichiers de type ext2, ext3 et ext4.
- Outils pour la gestion de base des systèmes de fichiers de type Btrfs, y compris les sous-volumes et les snapshots.
- Outils de gestion des systèmes de fichiers de type XFS.
- Connaissance des principes généraux des systèmes de fichiers de type ZFS.

b. Éléments mis en œuvre

- mkfs (mkfs.*)
- mkswap
- fsck (fsck.*)
- tune2fs, dumpe2fs et debugfs
- btrfs, btrfs-convert
- xfs_info, xfs_check, xfs_repair, xfsdump et xfsrestore
- smartd, smartctl

3. Création et configuration de systèmes de fichiers optionnels

Poids	2
Objectifs	Configuration de l'automontage de systèmes de fichiers de périphériques ou via le réseau, par autofs. Création de systèmes de fichiers pour les CD-ROM et autres périphériques. Notions de système de fichiers chiffré.

a. Compétences principales

- Fichiers de configuration d'autofs.
- Unités automount.
- Outils UDF et ISO 9660.
- Autres systèmes de fichiers pour CD-ROM (HFS).
- Extensions pour les systèmes de fichiers CD-ROM (Joliet, Rock Ridge, El Torito).
- Notions de base du chiffrement de données (dm-crypt, LUKS).

b. Éléments mis en œuvre

- /etc/auto.master
- /etc/auto.[dir]
- mkisofs
- cryptsetup

B. Administration du système de fichiers Linux

Pour stocker des fichiers et les mettre à disposition des applications, le système Linux utilise une arborescence globale, organisée en répertoires qui peuvent contenir d'autres répertoires et/ou des fichiers.

Les applications peuvent accéder à l'ensemble des fichiers du système de fichiers Linux, sous réserve du contrôle d'accès, sans avoir à connaître l'organisation physique du stockage de ces fichiers (sur un ou plusieurs disques durs, une ou plusieurs partitions de disques, des volumes logiques, sur la machine locale ou une machine distante, etc.).

L'administrateur configure l'organisation de l'arborescence globale du système de fichiers Linux, en utilisant un ou plusieurs systèmes de fichiers pour constituer cette arborescence.

Certains espaces de stockage sont en permanence accessibles à travers l'arborescence, d'autres sont amovibles et peuvent être rendus accessibles dynamiquement, par une commande du système ou de façon automatique (automontage).

Il est possible d'inclure dans l'arborescence globale du système de fichiers Linux des espaces de stockage se trouvant sur d'autres systèmes.

D'autre part, une partie de l'espace physique de stockage des informations doit être réservée pour les opérations de swap de la mémoire vive.

Le sujet 203.1 a pour objectif de s'assurer que le candidat a la maîtrise des opérations de configuration et de gestion du système de fichiers Linux, en combinant des systèmes de fichiers de types différents, permanents ou amovibles, éventuellement chiffrés, ainsi que de la gestion du swap.

1. Composants du système de fichiers Linux

L'arborescence du système de fichiers Linux a un répertoire de départ, le répertoire **racine** (*root*), noté /. Elle est constituée d'un ou plusieurs systèmes de fichiers autonomes, chacun monté sur un des répertoires de l'arborescence.

Un système de fichiers permet de structurer un espace de stockage, sous forme de fichiers et de répertoires. Cette organisation interne à l'espace de stockage est rendue accessible aux applications, sous forme de répertoires et de fichiers, par l'opération de **montage** sur un répertoire de l'arborescence globale.

Un système de fichiers non monté est vu comme un ensemble d'octets non structuré, accessible comme un tout, sans qu'il soit possible d'utiliser son organisation en répertoires et fichiers.

Les systèmes de fichiers peuvent être montés au démarrage du système sur l'arborescence globale du système de fichiers, ou pendant son fonctionnement, manuellement ou automatiquement.

Il existe différents types de systèmes de fichiers, qui peuvent être combinés entre eux au sein de l'arborescence globale du système de fichiers Linux.

2. Systèmes de fichiers physiques

Ces systèmes de fichiers sont associés à de l'espace de stockage physique. Ils sont organisés en répertoires et fichiers, qui sont stockés sur divers supports physiques : partitions de disques durs, volumes logiques, périphériques amovibles (CD-ROM, clé USB, etc.), espaces disques gérés par des systèmes distants (NFS, CIFS, SAN, NAS, etc.).

☞*Il existe de nombreux types de systèmes de fichiers physiques supportés par Linux, en particulier ext2, ext3, ext4, XFS, Btrfs et ZFS, qui seront décrits dans le cadre du sujet 203.2.*

3. Systèmes de fichiers virtuels

Certains types de systèmes de fichiers sont dits **virtuels**, car ils ne sont pas associés à un espace de stockage, mais sont directement gérés en mémoire vive. Ils doivent être montés pour donner accès à leurs répertoires et fichiers, mais leur durée de vie est limitée à leur période de montage : quand ils sont démontés (explicitement ou suite à l'arrêt du système), leur contenu est perdu.

Le type de système de fichiers virtuel tmpfs est conçu pour permettre de gérer en mémoire vive les fichiers temporaires. Il est utilisé par défaut par certaines distributions pour le contenu du répertoire `/tmp`. Ce mécanisme permet d'optimiser les temps d'accès pour ces fichiers qui n'ont pas vocation à être stockés de façon permanente.

Linux propose deux systèmes de fichiers virtuels standards, **proc** et **sys**, dont le but est de faciliter la communication avec le noyau. Ce dernier expose dynamiquement, sous forme de fichiers, des éléments de sa configuration et des objets qu'il gère (processus, fichiers, etc.). Ces deux systèmes de fichiers sont montés par défaut sur les répertoires `/proc` et `/sys` de l'arborescence globale du système de fichiers.

a. Le système de fichiers virtuel proc

Ce système de fichiers virtuel, de type **proc**, est monté par défaut sous le répertoire `/proc`. Il est géré dynamiquement par le noyau pour permettre de suivre l'état des processus actifs. Chaque processus est associé à un répertoire dont le nom est l'identifiant du processus. Quand le processus se termine, le répertoire est supprimé. Ce mécanisme permet de suivre les différents attributs des processus du système.

Ce système de fichiers virtuel donne également accès à de nombreuses tables de contrôle du noyau : partitions, montages, utilisation de la mémoire, etc. Il fournit également des informations concernant les caractéristiques du matériel pris en charge par le noyau.

Exemples

On lance l'éditeur Vim depuis une connexion SSH et on visualise les caractéristiques du processus ainsi créé depuis une autre connexion, dans le répertoire `/proc` *:*

```
vim /etc/hosts
```

Depuis un autre terminal :

```
ps -ef | grep vim
pba        14406   14300  0 16:08 pts/0    00:00:00 vi /etc/hosts
```

On cherche le répertoire correspondant au processus de PID 14406, dans `/proc` *:*

```
ls -ld /proc/14406
dr-xr-xr-x 9 pba pba 0  1 août  16:08 /proc/14406
```

Contenu du répertoire :

ls /proc/14406

```
arch_status          limits          root
attr                 loginuid        sched
autogroup            map_files       schedstat
auxv                 maps            sessionid
cgroup               mem             setgroups
clear_refs           mountinfo       smaps
cmdline              mounts          smaps_rollup
comm                 mountstats      stack
coredump_filter      net             stat
cpu_resctrl_groups   ns              statm
cpuset               numa_maps       status
cwd                  oom_adj         syscall
environ              oom_score       task
exe                  oom_score_adj   timens_offsets
fd                   pagemap         timers
fdinfo               patch_state     timerslack_ns
gid_map              personality     uid_map
io                   projid_map      wchan
```

Tous ces fichiers contiennent des informations sur le processus exécutant vi. Par exemple, on peut lire la ligne de commande exécutée par le processus :

cd /proc/14406
cat cmdline

```
vi/etc/hosts
```

On peut connaître le répertoire courant du processus :

ls -ld cwd

```
lrwxrwxrwx 1 pba pba 0  1 août  16:10 cwd -> /home/pba
```

cwd est un lien symbolique vers le répertoire courant du processus.

On sort du répertoire associé au processus et on quitte vi dans l'autre terminal :

cd ..
ls 14406

```
ls: impossible d'accéder à '14406': Aucun fichier ou dossier de ce type
```

Dès que le processus est terminé, le noyau supprime le répertoire correspondant dans le système de fichiers virtuel proc.

Dans le système de fichiers virtuel proc, on dispose de fichiers et de répertoires concernant différents éléments matériels et logiciels gérés par le noyau :

Pour connaître la version du noyau :

cat version

```
Linux version 5.10.0-13-amd64 (debian-kernel@lists.debian.org) (gcc-10
(Debian 10.2.1-6) 10.2.1 20210110, GNU ld (GNU Binutils for Debian) 2.35.2)
#1 SMP Debian 5.10.106-1 (2022-03-17)
```

Pour les informations concernant le (ou les) processeur(s) :

```
cat cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 104
model name      : AMD Athlon(tm) 64 X2 Dual-Core Processor TK-53tel(R)
Core(TM) i3-7020U CPU @ 2.30 GHz
[...]
processor       : 1
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 104
model name      : AMD Athlon(tm) 64 X2 Dual-Core Processor TK-53
```

b. Le système de fichiers virtuel sys

Ce système de fichiers virtuel, de type **sysfs**, est monté par défaut sous le répertoire `/sys`. Il fournit des informations concernant les caractéristiques des différents périphériques. Certains des pseudo-fichiers sont accessibles en écriture (sous contrainte des droits d'accès) et permettent de modifier dynamiquement des paramètres.

Exemple

Sur une distribution Debian :

```
ls /sys
block bus class dev devices firmware fs hypervisor kernel module power
```

Ces répertoires contiennent différents répertoires et fichiers, permettant de connaître les différents éléments matériels et logiciels gérés par le noyau.

```
ls /sys/dev/block
11:0  254:0  254:1  254:2  254:3  254:4  8:0  8:1  8:2  8:5
```

Chaque répertoire de `/sys/dev/block` correspond à un périphérique associé à un fichier spécial en mode bloc. Le nom est composé du numéro de majeur et du numéro de mineur du périphérique :

```
cd /sys/dev/block/11:0
cat device/model
DVDRW   DVR-K17
```

Le fichier spécial de majeur `11` et de mineur `0` est associé au lecteur/graveur de DVD de la machine.

4. Identification des systèmes de fichiers

L'identification des systèmes de fichiers peut se faire de différentes manières : fichier spécial associé, chemin d'accès réseau, chemin d'accès de fichier, label, UUID.

a. Fichier spécial en mode bloc

C'est la méthode traditionnelle Unix, reprise dans Linux. Elle permet d'identifier de façon unique le système de fichiers contenu dans l'espace de stockage associé au fichier spécial en mode bloc.