

# Linux Essentials

Version 1.6  
Français

010

# Table of Contents

<b>SUJET 1 : COMMUNAUTÉ LINUX ET CARRIÈRE DANS LE LOGICIEL LIBRE</b>	<b>1</b>
<b>1.1 Évolution de Linux et systèmes d'exploitation populaires</b>	<b>2</b>
1.1 Leçon 1	3
Introduction	3
Distributions	4
Systèmes embarqués	5
Linux et le Cloud	7
Exercices guidés	8
Exercices d'exploration	9
Résumé	10
Réponses aux exercices guidés	11
Réponses aux exercices d'exploration	13
<b>1.2 Applications libres majeures</b>	<b>14</b>
1.2 Leçon 1	15
Introduction	15
Paquets Logiciels	15
Installation des Paquets	16
Suppression des Paquets	19
Les Applications de Bureautique	21
Navigateurs Web	21
Multimédia	22
Programmes Serveurs	23
Partage des Données	24
Administration du Réseau	25
Langages de Programmation	26
Exercices Guidés	29
Exercices d'Exploration	31
Résumé	32
Réponses aux Exercices Guidés	33
Réponses aux Exercices d'Exploration	35
<b>1.3 Logiciel à code source ouvert et licences</b>	<b>36</b>
1.3 Leçon 1	37
Introduction	37
Définition des Logiciels Libres et Open Source	37
Licences	40
Les Modèles Commerciaux dans l'Open Source	44
Exercices guidés	47
Exercices d'exploration	48

Résumé .....	49
Réponses aux exercices guidés .....	50
Réponses aux exercices d'exploration .....	51
<b>1.4 Compétences informatiques et travail sous Linux .....</b>	<b>53</b>
1.4 Leçon 1 .....	54
Introduction .....	54
Interfaces Utilisateurs Linux .....	55
Utilisations de Linux par l'Industrie .....	57
Problèmes de Protection de la Vie Privée lors de l'Utilisation d'Internet .....	58
Chiffrement .....	62
Exercices Guidés .....	66
Exercices d'Exploration .....	68
Résumé .....	69
Réponses aux Exercices Guidés .....	70
Réponses aux Exercices d'Exploration .....	72
<b>SUJET 2 : TROUVER SON CHEMIN SUR UN SYSTÈME LINUX .....</b>	<b>73</b>
<b>2.1 Bases sur la ligne de commande .....</b>	<b>74</b>
2.1 Leçon 1 .....	75
Introduction .....	75
Structure de la Ligne de Commande .....	77
Types de Comportement des commandes .....	78
Insertion des guillemets (Quoting) .....	78
Exercices Guidés .....	82
Exercices d'Exploration .....	84
Résumé .....	85
Réponses aux Exercices Guidés .....	86
Réponses aux Exercices d'Exploration .....	88
2.1 Leçon 2 .....	89
Introduction .....	89
Les Variables .....	89
Manipuler des Variables .....	90
Exercices Guidés .....	95
Exercices d'Exploration .....	96
Résumé .....	97
Réponses aux Exercices Guidés .....	98
Réponses aux Exercices d'Exploration .....	99
<b>2.2 Utilisation de la ligne de commande pour obtenir de l'aide .....</b>	<b>101</b>
2.2 Leçon 1 .....	102
Introduction .....	102
Obtenir de l'Aide sur la Ligne de Commande .....	103

Localisation des fichiers .....	106
Exercices Guidés .....	108
Exercices d'Exploration .....	110
Résumé .....	111
Réponses aux Exercices Guidés .....	112
Réponses aux Exercices d'Exploration .....	115
<b>2.3 Utilisation des répertoires et liste des fichiers .....</b>	<b>117</b>
2.3 Leçon 1 .....	118
Introduction .....	118
Fichiers et Répertoires .....	118
Noms de Fichiers et de Répertoires .....	119
Naviguer dans le Système de Fichiers .....	119
Chemins Absolus et Relatifs .....	121
Exercices Guidés .....	123
Exercices d'Exploration .....	125
Résumé .....	126
Réponses aux Exercices Guidés .....	127
Réponses aux Exercices d'Exploration .....	130
2.3 Leçon 2 .....	131
Introduction .....	131
Répertoires Personnels .....	131
Le Chemin Relatif Spécial pour le Répertoire Personnel .....	133
Chemins des Fichiers Relatifs au Répertoire Personnel .....	134
Fichiers et Répertoires Cachés .....	135
L'Option de Listage Long .....	136
Options ls Supplémentaires .....	137
Récursion en Bash .....	138
Exercices Guidés .....	140
Exercices d'Exploration .....	142
Résumé .....	143
Réponses aux Exercices Guidés .....	144
Réponses aux Exercices d'Exploration .....	146
<b>2.4 Création, déplacement et suppression de fichiers .....</b>	<b>147</b>
2.4 Leçon 1 .....	148
Introduction .....	148
Sensibilité à la Casse .....	149
Création de Répertoires .....	149
Création de Fichiers .....	151
Renommage de Fichiers .....	152
Déplacement des Fichiers .....	153

Suppression de Fichiers et de Répertoires .....	154
Copie de Fichiers et de Répertoires .....	156
Globber .....	158
Exercices Guidés .....	163
Exercices d'Exploration .....	165
Résumé .....	166
Réponses aux Exercices Guidés .....	168
Réponses aux Exercices d'Exploration .....	171
<b>SUJET 3 : LE POUVOIR DE LA LIGNE DE COMMANDE .....</b>	<b>173</b>
<b>3.1 Archivage de fichiers en ligne de commande .....</b>	<b>174</b>
3.1 Leçon 1 .....	175
Introduction .....	175
Les Outils de Compression .....	176
Les Outils d'Archivage .....	179
La Gestion des fichiers ZIP .....	182
Exercices Guidés .....	184
Exercices d'Exploration .....	185
Résumé .....	186
Réponses aux Exercices Guidés .....	188
Réponses aux Exercices d'Exploration .....	190
<b>3.2 Recherche et extraction de données à partir de fichiers .....</b>	<b>191</b>
3.2 Leçon 1 .....	192
Introduction .....	192
Redirection des E/S .....	192
Les Pipes de la Ligne de Commande .....	197
Exercices Guidés .....	200
Exercices d'Exploration .....	201
Résumé .....	202
Réponses aux Exercices Guidés .....	203
Réponses aux Exercices d'Exploration .....	205
3.2 Leçon 2 .....	206
Introduction .....	206
Recherche dans les Fichiers avec grep .....	206
Les Expressions Régulières .....	207
Exercices Guidés .....	211
Exercices d'Exploration .....	212
Résumé .....	213
Réponses aux Exercices Guidés .....	214
Réponses aux Exercices d'Exploration .....	216
<b>3.3 Conversion de commandes en script .....</b>	<b>218</b>

3.3 Leçon 1 .....	219
Introduction .....	219
Affichage de la Sortie .....	219
Rendre un Script Exécutable .....	220
Les Commandes et PATH .....	220
Permission d'Exécution .....	221
Définir l'Interprète .....	222
Les Variables .....	224
L'Utilisation des Guillemets avec des Variables .....	225
Les Arguments .....	227
Retourner le Nombre d'Arguments .....	228
La Logique Conditionnelle .....	228
Exercices Guidés .....	231
Exercices d'Exploration .....	233
Résumé .....	234
Réponses aux Exercices Guidés .....	236
Réponses aux Exercices d'Exploration .....	238
3.3 Leçon 2 .....	240
Introduction .....	240
Codes de Sortie .....	241
Gérer Plusieurs Arguments .....	243
Les Boucles For .....	244
Usage d'Expressions Régulières pour Vérifier des Erreurs .....	247
Exercices Guidés .....	249
Exercices d'Exploration .....	251
Résumé .....	252
Réponses aux Exercices Guidés .....	253
Réponses aux Exercices d'Exploration .....	255
<b>SUJET 4 : LE SYSTÈME D'EXPLOITATION LINUX .....</b>	<b>256</b>
<b>4.1 Choix d'un système d'exploitation .....</b>	<b>257</b>
4.1 Leçon 1 .....	258
Introduction .....	258
Qu'est-ce qu'un Système d'Exploitation .....	258
Choisir une Distribution Linux .....	259
Systèmes d'Exploitation Non Linux .....	263
Exercices Guidés .....	266
Exercices d'Exploration .....	268
Résumé .....	269
Réponses aux Exercices Guidés .....	270
Réponses aux Exercices d'Exploration .....	272

<b>4.2 Compréhension du matériel informatique</b>	<b>273</b>
4.2 Leçon 1	274
Introduction	274
Les Alimentations Électriques	275
La Carte Mère	275
La Mémoire	276
Les Processeurs	277
Le stockage	280
Les partitions	282
Les périphériques	282
Les Pilotes et Fichiers de Périphériques	283
Exercices Guidés	285
Exercices d'Exploration	286
Résumé	287
Réponses aux Exercices Guidés	288
Réponses aux Exercices d'Exploration	290
<b>4.3 Localisation des données</b>	<b>291</b>
4.3 Leçon 1	292
Introduction	292
Les Programmes et leurs Configurations	293
Le Noyau Linux	297
Les Périphériques Matériels	300
Mémoire et Types de Mémoire	302
Exercices Guidés	305
Exercices d'Exploration	307
Résumé	308
Réponses aux Exercices Guidés	310
Réponses aux Exercices d'Exploration	312
4.3 Leçon 2	313
Introduction	313
Les Processus	313
Journalisation du Système et Messagerie du Système	317
Exercices Guidés	323
Exercices d'Exploration	326
Résumé	328
Réponses aux Exercices Guidés	330
Réponses aux Exercices d'Exploration	334
<b>4.4 Intégration au réseau</b>	<b>336</b>
4.4 Leçon 1	337
Introduction	337

Réseau de Couche de Liaison .....	338
Réseau IPv4 .....	339
Réseau IPv6 .....	344
DNS .....	348
Les Sockets .....	349
Exercices Guidés .....	352
Exercices d'Exploration .....	353
Résumé .....	354
Réponses aux Exercices Guidés .....	355
Réponses aux Exercices d'Exploration .....	356
<b>SUJET 5 : SÉCURITÉ ET DROITS D'ACCÈS AUX FICHIERS .....</b>	<b>358</b>
<b>5.1 Sécurité élémentaire et identification des catégories d'utilisateurs .....</b>	<b>359</b>
5.1 Leçon 1 .....	360
Introduction .....	360
Les Comptes .....	361
Obtenir des Informations sur vos Utilisateurs .....	364
Changement d'Utilisateur et Escalade des Privilèges .....	366
Les Fichiers de Contrôle d'Accès .....	368
Exercices Guidés .....	376
Exercices d'Exploration .....	378
Résumé .....	380
Réponses aux Exercices Guidés .....	382
Réponses aux Exercices d'Exploration .....	384
<b>5.2 Création des utilisateurs et des groupes .....</b>	<b>386</b>
5.2 Leçon 1 .....	387
Introduction .....	387
Le Fichier /etc/passwd .....	388
Le Fichier /etc/group .....	389
Le Fichier /etc/shadow .....	389
Le Fichier /etc/gshadow .....	390
Ajout et Suppression de Comptes Utilisateurs .....	391
Le Répertoire Squelette .....	394
Ajout et Suppression de Groupes .....	394
La Commande passwd .....	395
Exercices Guidés .....	396
Exercices d'Exploration .....	398
Résumé .....	399
Réponses aux Exercices Guidés .....	400
Réponses aux Exercices d'Exploration .....	402
<b>5.3 Gestion des propriétés et des droits d'accès aux fichiers .....</b>	<b>405</b>



5.3 Leçon 1 .....	406
Introduction.....	406
Recherche d'Informations sur les Fichiers et les Répertoires .....	406
Et les Répertoires ? .....	408
Voir les Fichiers Cachés .....	409
Comprendre les Types de Fichiers.....	409
Comprendre les Permissions .....	410
Modification des Permissions des Fichiers.....	412
Mode Symbolique.....	413
Mode Numérique .....	415
Modification de la Propriété des Fichiers .....	416
Interroger les Groupes .....	417
Permissions Spéciales .....	417
Exercices Guidés .....	421
Exercices d'Exploration .....	423
Résumé.....	424
Réponses aux Exercices Guidés .....	425
Réponses aux Exercices d'Exploration .....	428
<b>5.4 Répertoires et fichiers spéciaux .....</b>	<b>432</b>
5.4 Leçon 1 .....	433
Introduction.....	433
Fichiers Temporaires .....	433
Comprendre les Liens .....	435
Exercices Guidés .....	440
Exercices d'Exploration .....	441
Résumé.....	444
Réponses aux Exercices Guidés .....	445
Réponses aux Exercices d'Exploration .....	446
<b>Impression .....</b>	<b>450</b>



**Linux  
Professional  
Institute**

## **Sujet 1 : Communauté Linux et carrière dans le logiciel libre**



## 1.1 Évolution de Linux et systèmes d'exploitation populaires

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 1.1

### Valeur

2

### Domaines de connaissance les plus importants

- Distributions
- Systèmes embarqués
- Linux dans le nuage

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Debian, Ubuntu (LTS)
- CentOS, openSUSE, Red Hat, SUSE
- Linux Mint, Scientific Linux
- Raspberry Pi, Raspbian
- Android



# 1.1 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	1 La Communauté Linux et une Carrière dans l'Open Source
<b>Objectif :</b>	1.1 Évolution de Linux et les Systèmes d'Exploitation Populaires
<b>Leçon :</b>	1 sur 1

## Introduction

Linux est l'un des systèmes d'exploitation les plus populaires. Son développement a été initié par Linus Torvalds en 1991. Le système d'exploitation s'est inspiré de Unix, un autre système d'exploitation développé dans les années 1970 par les laboratoires AT&T. Unix était adapté aux petits ordinateurs. À l'époque, les “petits” ordinateurs étaient considérés comme des machines qui n'avaient pas besoin d'une salle entière avec air conditionné et qui coûtaient moins d'un million de dollars. Plus tard, ils ont été considérés comme des machines pouvant être soulevées par deux personnes. À cette époque, Unix n'était pas disponible sur les petits ordinateurs comme les ordinateurs de bureau basés sur la plate-forme x86. C'est pourquoi Linus Torvalds, qui était étudiant à cette époque, a commencé à réaliser un système d'exploitation de type Unix qui était censé fonctionner sur cette plate-forme.

La plupart du temps, Linux utilise les mêmes principes et idées que base d'Unix, mais Linux lui-même ne contient pas de code Unix, car c'est un projet indépendant. Linux n'est pas soutenu par une entreprise individuelle, mais par une communauté internationale de programmeurs. Comme il est librement disponible, il peut être utilisé par n'importe qui sans restrictions.

## Distributions

Une *distribution* Linux est un ensemble qui se compose d'un *noyau* Linux et d'une sélection d'applications qui sont maintenues par une entreprise ou une communauté d'utilisateurs. L'objectif d'une distribution est d'optimiser le noyau et les applications qui sont exécutées sur le système d'exploitation pour un certain usage ou groupe d'utilisateurs. Les distributions comprennent souvent des outils qui leur sont spécifiques pour l'installation de logiciels et l'administration du système. C'est pourquoi certaines distributions sont principalement utilisées pour des environnements de bureau où elles doivent être faciles à utiliser, tandis que d'autres sont principalement utilisées pour fonctionner sur des serveurs afin d'utiliser les ressources disponibles de la manière la plus performante possible.

Une autre façon de classer les distributions est de se référer à la *famille de distribution* à laquelle elles appartiennent. Les distributions de la famille Debian utilisent le gestionnaire de paquets `dpkg` pour gérer les logiciels qui sont exécutés sur le système d'exploitation. Les paquets qui peuvent être installés avec le gestionnaire de paquets sont maintenus par des membres volontaires de la communauté de la distribution. Les mainteneurs utilisent le format de paquet `deb` pour spécifier comment le logiciel est installé sur le système d'exploitation et comment il est configuré par défaut. Tout comme une distribution, un paquet est un ensemble composé d'un logiciel avec la configuration et la documentation correspondantes qui facilitent l'installation, la mise à jour et l'utilisation du logiciel par l'utilisateur.

La distribution *Debian GNU/Linux* est la plus grande distribution de la famille des distributions Debian. Le projet Debian GNU/Linux a été lancé par Ian Murdock en 1993. Aujourd'hui, des milliers de bénévoles travaillent sur le projet. Debian GNU/Linux vise à fournir un système d'exploitation très fiable. Elle promeut également la vision de Richard Stallman d'un système d'exploitation qui respecte les libertés de l'utilisateur d'exécuter, d'étudier, de distribuer et d'améliorer les logiciels. C'est pourquoi elle ne fournit aucun logiciel propriétaire par défaut.

*Ubuntu* est une autre distribution basée sur Debian qui mérite d'être mentionnée. Ubuntu a été créée par Mark Shuttleworth et son équipe en 2004, avec pour mission d'apporter un environnement de bureau Linux facile à utiliser. La mission d'Ubuntu est de fournir des logiciels libres à tous dans le monde entier ainsi que de réduire le coût des services professionnels. La distribution a une sortie prévue tous les six mois, avec une sortie à support à long terme tous les deux ans.

*Red Hat* est une distribution Linux développée et maintenue par l'entreprise de logiciels du même nom, qui a été rachetée par IBM en 2019. La distribution Red Hat Linux en 1994 et rebaptisée en 2003 *Red Hat Enterprise Linux*, souvent abrégée en RHEL. Elle est offerte aux entreprises en tant que solution d'entreprise fiable, son soutien est fourni par Red Hat et inclus des logiciels visant à faciliter l'utilisation de Linux dans les environnements de serveurs professionnels. Certains de ses

composants nécessitent des abonnements ou des licences payantes. Le projet *CentOS* utilise le code source librement disponible de Red Hat Enterprise Linux et le compile en une distribution qui est disponible entièrement gratuitement, mais en retour n'est accompagnée d'aucun support commercial.

RHEL et CentOS sont toutes deux optimisées pour une utilisation dans des environnements de serveurs. Le projet *Fedora* a été fondé en 2003 et crée une distribution Linux destinée aux ordinateurs de bureau. Red Hat a initié et maintient la distribution Fedora depuis lors. Fedora est très innovante et adopte très rapidement les nouvelles technologies. Elle est parfois considérée comme un banc d'essai pour les nouvelles technologies qui pourraient plus tard être incluses dans RHEL. Toutes les distributions basées sur Red Hat utilisent le format de paquet `rpm`.

La société SUSE a été fondée en 1992 en Allemagne en tant que prestataire de services Unix. La première version de *SUSE Linux* a été publiée en 1994. Au fil des ans, SUSE Linux est devenu surtout connue pour son outil de configuration YaST. Cet outil permet aux administrateurs d'installer et de configurer des logiciels et du matériel, et de mettre en place des serveurs et des réseaux. Tout comme RHEL, SUSE publie *SUSE Linux Enterprise Server*, qui est sa version commerciale. Cette version est moins fréquemment publiée et convient au déploiement en entreprise et en production. Elle est distribuée sous forme de serveur ainsi que d'environnement de bureau, avec des paquets adaptés. En 2004, SUSE a lancé le projet *openSUSE*, qui a offert de nouvelles opportunités aux développeurs et aux utilisateurs de tester et de développer encore plus le système. La distribution openSUSE est proposée en téléchargement gratuit.

Des distributions indépendantes ont été publiées au fil des ans. Certaines d'entre elles sont basées sur Red Hat ou Ubuntu, d'autres sont conçues pour améliorer une propriété spécifique d'un système ou d'un matériel. Il existe des distributions construites avec des fonctionnalités spécifiques comme *QubesOS*, un environnement de bureau très sécurisé, ou *Kali Linux*, qui fournit un environnement permettant d'exploiter les vulnérabilités des logiciels, principalement utilisé par les testeurs de pénétration. Depuis peu de temps, on trouve des distributions Linux variées de très petite taille pour fonctionner spécifiquement dans des conteneurs Linux, comme Docker. Il existe également des distributions conçues spécifiquement pour les composants de systèmes embarqués et même pour les appareils intelligents.

## Systèmes embarqués

Les systèmes embarqués sont une combinaison de matériel informatique et de logiciels conçus pour avoir une fonction spécifique au sein d'un système plus large. Ils font généralement partie d'autres dispositifs et aident à contrôler ces dispositifs. On trouve des systèmes embarqués dans des applications automobiles, médicales et même militaires. En raison de sa grande variété d'applications, une variété de systèmes d'exploitation basés sur le noyau Linux a été développée afin d'être utilisée dans les systèmes embarqués. Une part importante des appareils intelligents

sont équipés d'un système d'exploitation basé sur le noyau Linux.

Par conséquent, les systèmes embarqués sont équipés de logiciels embarqués. Le but de ces logiciels est d'accéder au matériel et de le rendre utilisable. Les principaux avantages de Linux par rapport à tout logiciel embarqué propriétaire sont la compatibilité entre les plates-formes des différents fournisseurs, le développement, l'assistance et l'absence de frais de licence. Deux des projets de logiciels embarqués les plus populaires sont Android, principalement utilisé sur les téléphones mobiles par divers fournisseurs, et Raspbian, principalement utilisé sur Raspberry Pi.

## Android

Android est principalement un système d'exploitation mobile développé par Google. Android Inc. a été fondé en 2003 à Palo Alto, en Californie. La société a initialement créé un système d'exploitation destiné à fonctionner avec des appareils photo numériques. En 2005, Google a racheté Android Inc. et a développé son système pour en faire l'un des plus grands systèmes d'exploitation mobiles.

La base d'Android est une version modifiée du noyau Linux avec des logiciels open source supplémentaires. Le système d'exploitation est principalement développé pour les appareils à écran tactile, mais Google a également développé des versions pour la télévision et les montres connectées. Différentes versions d'Android ont été développées pour les consoles de jeux, les appareils photo numériques, ainsi que pour les PC.

Android est disponible gratuitement en open source sous le nom de *Android Open Source Project* (AOSP). Google propose une série de composants propriétaires en plus du noyau open source d'Android. Ces composants comprennent des applications telles que Google Agenda, Google Maps, Google Mail, le navigateur Chrome ainsi que le Google Play Store qui facilite l'installation des applications. La plupart des utilisateurs considèrent ces outils comme faisant partie intégrante de leur expérience Android. Par conséquent, presque tous les appareils mobiles livrés avec Android en Europe et en Amérique comprennent des logiciels propriétaires de Google.

Android sur les appareils embarqués présente de nombreux avantages. Le système d'exploitation est intuitif et facile à utiliser avec une interface utilisateur graphique, il dispose d'une très large communauté de développeurs, il est donc facile de trouver de l'aide pour le développement. Il a également du soutien fourni par la majorité des fournisseurs de matériel avec un pilote Android, il est donc facile et rentable de prototyper un système entier.

## Raspbian et le Raspberry Pi

Le Raspberry Pi est un ordinateur à bas prix, de la taille d'une carte de crédit, qui peut fonctionner comme un ordinateur de bureau à part entière, mais qui peut aussi être utilisé dans

un système Linux intégré. Il est développé par la Fondation Raspberry Pi, qui est une organisation caritative à vocation éducative basée au Royaume-Uni. Il a principalement pour but d'apprendre aux jeunes à programmer et à comprendre les fonctionnalités des ordinateurs. Le Raspberry Pi peut être conçu et programmé pour effectuer des tâches ou des opérations souhaitées qui font partie d'un système beaucoup plus complexe.

Les spécialités du Raspberry Pi incluent un ensemble de broches GPIO (General Purpose Input-Output) qui peuvent être utilisées pour connecter des appareils électroniques et des cartes d'extension. Cela permet d'utiliser le Raspberry Pi comme une plate-forme pour le développement de matériel. Bien qu'il était destiné à des fins éducatives, le Raspberry Pi est aujourd'hui utilisé dans divers projets de bricolage ainsi que pour le prototypage industriel lors du développement de systèmes embarqués.

Le Raspberry Pi utilise des processeurs ARM. Divers systèmes d'exploitation, dont Linux, fonctionnent sur le Raspberry Pi. Comme le Raspberry Pi n'est pas équipé d'un disque dur, le système d'exploitation est lancé à partir d'une carte mémoire SD. L'une des distributions Linux les plus connues pour le Raspberry Pi est *Raspbian*. Comme son nom l'indique, elle appartient à la famille des distributions Debian. Elle est personnalisée pour être installée sur le matériel Raspberry Pi et fournit plus de 35 000 paquets optimisés pour cet environnement. Outre Raspbian, il existe de nombreuses autres distributions Linux pour Raspberry Pi, par exemple Kodi, qui transforme le Raspberry Pi en un centre multimédia.

## Linux et le Cloud

Le terme *cloud computing* fait référence à une façon standardisée de consommer les ressources informatiques, soit en les achetant à un fournisseur de cloud public, soit en exploitant un cloud privé. Selon les rapports de 2017, Linux exécute 90 % de la charge de travail du cloud public. Chaque fournisseur de cloud computing, d'*Amazon Web Services* (AWS) à *Google Cloud Platform* (GCP), propose différentes formes de Linux. Même Microsoft propose aujourd'hui des machines virtuelles basées sur Linux dans son cloud *Azure*.

Linux est généralement proposé dans le cadre d'offres *Infrastructure as a Service* (IaaS). Les instances IaaS sont des machines virtuelles qui sont fournies en quelques minutes dans le cloud. Lors du démarrage d'une instance IaaS, une image est choisie qui contient les données qui sont déployées dans la nouvelle instance. Les fournisseurs de cloud computing proposent diverses images contenant des installations prêtes à l'emploi des distributions Linux les plus courantes ainsi que de leurs propres versions de Linux. L'utilisateur du cloud choisit une image contenant sa distribution préférée et peut accéder à une instance de cloud exécutant cette distribution peu de temps après. La plupart des fournisseurs de services cloud ajoutent des outils à leurs images pour adapter l'installation à une instance de cloud spécifique. Ces outils peuvent, par exemple, étendre les systèmes de fichiers de l'image pour qu'elle s'adapte au disque dur réel de la machine virtuelle.



## Exercices guidés

1. En quoi Debian GNU/Linux est-elle différente d'Ubuntu ? Citez deux aspects.

2. Quels sont les environnements/plateformes les plus courants pour lesquels Linux est utilisé ? Citez trois environnements/plateformes différents et indiquez une distribution que vous pouvez utiliser pour chacun d'eux.

3. Vous envisagez d'installer une distribution Linux dans un nouvel environnement. Citez quatre éléments à prendre en compte lors du choix d'une distribution.

4. Citez trois appareils sur lesquels fonctionne le système d'exploitation Android, autres que les smartphones.

5. Expliquez trois principaux avantages du cloud computing.

## Exercices d'exploration

1. Compte tenu des coûts et des performances, quelles sont les distributions qui conviennent le mieux à une entreprise qui vise à réduire les coûts de licence tout en maintenant ses performances au plus haut niveau ? Expliquez pourquoi.

2. Quels sont les principaux avantages du Raspberry Pi et quelles fonctions peut-il assumer en entreprise ?

3. Quelle est la gamme de distributions offertes par Amazon Cloud Services et Google Cloud ? Citez au moins trois distributions communes et deux distributions différentes.

# Résumé

Dans cette leçon vous avez appris :

- Quelles sont les distributions de Linux
- Quels sont les systèmes Linux embarqués
- Comment sont utilisés les systèmes Linux embarqués
- Les différents usages d'Android
- Les différents usages d'un Raspberry Pi
- Ce qu'est le Cloud Computing
- Quel rôle joue Linux dans le cloud computing

# Réponses aux exercices guidés

1. En quoi Debian GNU/Linux est-il différent d'Ubuntu ? Citez deux aspects.

Ubuntu est basée sur une image de Debian, il y a donc de nombreuses similitudes. Cependant, il existe encore des différences significatives entre elles. La première est la simplicité pour les débutants. Ubuntu est recommandé pour les débutants en raison de sa facilité d'utilisation alors que Debian est recommandé pour les utilisateurs plus avancés. La principale différence est la complexité de la configuration utilisateur que Ubuntu enlève pendant le processus d'installation.

Une autre différence est la stabilité de chaque distribution. Debian est considérée comme plus stable que Ubuntu. Cela s'explique par le fait que Debian reçoit des mises à jours testées en détail, mais en plus petit nombre, qui sont testées en détail et que l'ensemble du système d'exploitation est plus stable. D'autre part, Ubuntu permet à l'utilisateur d'utiliser les dernières versions des logiciels et toutes les nouvelles technologies.

2. Quels sont les environnements/plateformes les plus courants pour lesquels Linux est utilisé ? Citez trois environnements/plateformes différents et indiquez une distribution que vous pouvez utiliser pour chacun d'eux.

Quelques-uns des environnements/plateformes les plus courants sont les smartphones, les ordinateurs de bureau et les serveurs. Sur les smartphones, il peut être utilisé par des distributions telles qu'Android. Sur les ordinateurs de bureau et les serveurs, il peut être utilisé par toute distribution qui convient le mieux aux fonctionnalités de cette machine, de Debian, Ubuntu à CentOS et Red Hat Enterprise Linux.

3. Vous envisagez d'installer une distribution Linux dans un nouvel environnement. Citez quatre éléments à prendre en compte lors du choix d'une distribution.

Lors du choix d'une distribution, il faut tenir compte du coût, des performances, de l'évolutivité, de la stabilité et des exigences matérielle du système.

4. Citez trois appareils sur lesquels fonctionne le système d'exploitation Android, autres que les smartphones.

Les autres appareils sur lesquels fonctionne Android sont les téléviseurs intelligents, les tablettes informatiques, Android Auto et les montres intelligentes.

5. Expliquez les trois principaux avantages du cloud computing.

Les principaux avantages du cloud computing sont la flexibilité, la facilité de récupération et le

faible coût d'utilisation. Les services basés sur le cloud computing sont faciles à mettre en œuvre et à étendre, en fonction des besoins de l'entreprise. Il présente un avantage majeur dans les solutions de sauvegarde et de récupération, car il permet aux entreprises de se remettre des incidents plus rapidement et avec moins de répercussions. En outre, il réduit les coûts d'exploitation, car il permet de payer uniquement les ressources utilisées par une entreprise, sur la base d'un modèle d'abonnement.

## Réponses aux exercices d'exploration

1. Compte tenu des coûts et des performances, quelles sont les distributions qui conviennent le mieux à une entreprise qui vise à réduire les coûts de licence tout en maintenant ses performances au plus haut niveau ? Expliquez pourquoi.

L'une des distributions les plus adaptées aux entreprises est CentOS. En effet, elle intègre tous les produits Red Hat, qui sont ensuite utilisés au sein de leur système d'exploitation commercial, tout en étant libre d'utilisation. De manière similaire, les versions LTS d'Ubuntu garantissent une prise en charge sur une plus longue période. Les versions stables de Debian GNU/Linux sont également souvent utilisées dans les environnements d'entreprise.

2. Quels sont les principaux avantages du Raspberry Pi et quelles fonctions peuvent-ils assumer dans les affaires ?

Le Raspberry Pi est de petite taille alors qu'il fonctionne comme un ordinateur normal. En outre, il est peu coûteux et peut gérer le trafic web et de nombreuses autres fonctionnalités. Il peut être utilisé comme serveur, pare-feu et peut servir de carte mère pour les robots et de nombreux autres petits dispositifs.

3. Quelle est la gamme de distributions offertes par Amazon Cloud Services et Google Cloud ? Citez au moins trois communes et deux différentes.

Les distributions communes entre Amazon et Google Cloud Services sont Ubuntu, CentOS et Red Hat Enterprise Linux. Chaque fournisseur de services cloud propose également des distributions spécifiques que l'autre ne propose pas. Amazon dispose d'Amazon Linux et de Kali Linux, tandis que Google propose l'utilisation de FreeBSD et de Windows Servers.



## 1.2 Applications libres majeures

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 1.2

### Valeur

2

### Domaines de connaissance les plus importants

- Applications pour postes de travail
- Applications pour serveurs
- Langages de programmation
- Outils de gestion et dépôts de paquetages

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Nextcloud, ownCloud
- Apache HTTPD, NGINX, MariaDB, MySQL, NFS, Samba
- C, Java, JavaScript, Perl, shell, Python, PHP
- dpkg, apt-get, rpm, yum



Linux  
Professional  
Institute

## 1.2 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	1 La communauté Linux et une Carrière dans l'Open Source
<b>Objectif :</b>	1.2 Les Principales Applications Open Source
<b>Leçon :</b>	1 sur 1

### Introduction

Une application est un programme informatique dont le but n'est pas directement lié au fonctionnement interne de l'ordinateur, mais aux tâches effectuées par l'utilisateur. Les distributions Linux offrent de nombreuses applications permettant d'effectuer diverses tâches, telles que des applications de bureau, des navigateurs web, des lecteurs et éditeurs multimédia, etc. Il existe souvent plusieurs applications ou outils pour effectuer une tâche particulière. Il appartient à l'utilisateur de choisir l'application qui répond le mieux à ses besoins.

### Paquets Logiciels

Presque toutes les distributions Linux proposent un ensemble d'applications préinstallées par défaut. Outre ces applications préinstallées, une distribution dispose d'un dépôt de paquets avec une vaste collection d'applications disponibles à l'installation via son *gestionnaire de paquets*. Bien que les différentes distributions proposent à peu près les mêmes applications, il existe plusieurs systèmes de gestion des paquets pour différentes distributions. Par exemple, Debian, Ubuntu et Linux Mint utilisent les outils `dpkg`, `apt-get` et `apt` pour installer des paquets logiciels, généralement appelés *paquets DEB*. Les distributions telles que Red Hat, Fedora et CentOS utilisent



plutôt les commandes `rpm`, `yum` et `dnf`, qui à leur tour installent des *paquets RPM*. Comme la gestion des paquets des applications est différente pour chaque famille de distribution, il est très important d'installer les paquets à partir du dépôt approprié conçu pour la distribution. L'utilisateur final n'a généralement pas à se soucier de ces détails, car le gestionnaire de paquets de la distribution choisira les bons paquets, les dépendances requises et les futures mises à jour. Les dépendances sont des paquets auxiliaires nécessaires aux programmes. Par exemple, si une bibliothèque fournit des fonctions permettant de traiter des images JPEG utilisées par plusieurs programmes, cette bibliothèque est probablement empaquetée dans son propre paquet dont dépendent toutes les applications utilisant la bibliothèque.

Les commandes `dpkg` et `rpm` fonctionnent sur des fichiers de paquets individuels. En pratique, presque toutes les tâches de gestion des paquets sont effectuées par les commandes `apt-get` ou `apt` sur les systèmes qui utilisent des paquets DEB ou par `yum` ou `dnf` sur les systèmes qui utilisent des paquets RPM. Ces commandes fonctionnent avec des catalogues de paquets, peuvent télécharger de nouveaux paquets et leurs dépendances, et vérifier les nouvelles versions des paquets installés.

## Installation des Paquets

Supposons que vous ayez entendu parler d'une commande appelée `figlet` qui imprime un texte agrandi sur le terminal et que vous vouliez l'essayer. Cependant, vous obtenez le message suivant après avoir exécuté la commande `figlet` :

```
$ figlet
-bash: figlet: command not found
```

Cela signifie probablement que le paquet n'est pas installé sur votre système. Si votre distribution fonctionne avec des paquets DEB, vous pouvez effectuer une recherche dans ses dépôts en utilisant `apt-cache search nom_du_paquet` ou `apt search nom_du_paquet`. La commande `apt-cache` est utilisée pour rechercher des paquets et pour lister les informations sur les paquets disponibles. La commande suivante recherche toute occurrence du terme "figlet" dans les noms et les descriptions des paquets :

```
$ apt-cache search figlet
figlet - Make large character ASCII banners out of ordinary text
```

La recherche a permis d'identifier un paquet appelé *figlet* qui correspond à la commande manquante. L'installation et la suppression d'un paquet nécessitent des autorisations spéciales accordées uniquement à l'administrateur du système : l'utilisateur nommé `root`. Sur les systèmes

de bureau, les utilisateurs ordinaires peuvent installer ou supprimer des paquets en ajoutant la commande `sudo` aux commandes d'installation et de suppression. Pour ce faire, vous devrez taper votre mot de passe. Pour les paquets DEB, l'installation est effectuée avec la commande `apt-get install nom_du_paquet` ou `apt install nom_du_paquet` :

```
$ sudo apt-get install figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

À ce stade, le paquet sera téléchargé et installé sur le système. Toutes les dépendances dont le paquet aura éventuellement besoin seront également téléchargées et installées :

```
Need to get 184 kB of archives.
After this operation, 741 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian stretch/main armhf figlet armhf 2.2.5-2 [184 kB]
Fetched 184 kB in 0s (213 kB/s)
Selecting previously unselected package figlet.
(Reading database ... 115701 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_armhf.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in
auto mode
Processing triggers for man-db (2.7.6.1-2) ...
```

Une fois le téléchargement terminé, tous les fichiers sont copiés aux emplacements appropriés, toute configuration supplémentaire sera effectuée et la commande deviendra disponible :

```
$ figlet Awesome!
```

```

  _
 / \_      _ _ _ _ _ _ _ _ _ _
/_ _ \ \ / \ / / _ \ \ _ \| ' _ \ / \ \
/_ _ _ \ V V / _ \ \ _ \ ( ) | | | | | _/_
/_/_ _ \ \ \ \ \/_/_ \ _ \| _ \/_/_ \| | | | | \ _ \ ( )

```

Dans les distributions basées sur des paquets RPM, les recherches sont effectuées en utilisant yum

`search nom_du_paquet` ou `dnf search nom_du_paquet`. Disons que vous voulez afficher un texte de manière plus irrévérencieuse, suivi d'une vache caricaturale, mais que vous n'êtes pas sûr du paquet qui peut effectuer cette tâche. Comme pour les paquets DEB, les commandes de recherche RPM acceptent des termes descriptifs :

```
$ yum search speaking cow
```

```
Last metadata expiration check: 1:30:49 ago on Tue 23 Apr 2019 11:02:33 PM -03.  
===== Name & Summary Matched: speaking, cow =====  
cowsay.noarch : Configurable speaking/thinking cow
```

Après avoir trouvé un paquet approprié dans le dépôt, il peut être installé avec `yum install nom_du_paquet` ou `dnf install nom_du_paquet` :

```
$ sudo yum install cowsay
```

```
Last metadata expiration check: 2:41:02 ago on Tue 23 Apr 2019 11:02:33 PM -03.  
Dependencies resolved.
```

```
=====
```

Package	Arch	Version	Repository	Size
Installing:				
cowsay	noarch	3.04-10.fc28	fedora	46 k

```
=====
```

```
Transaction Summary
```

```
=====
```

Install	1 Package
---------	-----------

```
Total download size: 46 k
```

```
Installed size: 76 k
```

```
Is this ok [y/N]: y
```

Une fois de plus, le paquet souhaité et toutes ses dépendances possibles seront téléchargés et installés :

```
Downloading Packages:
```

```
cowsay-3.04-10.fc28.noarch.rpm          490 kB/s | 46 kB    00:00
```

```
=====
```

Total	53 kB/s   46 kB	00:00
-------	-----------------	-------

```
Running transaction check
```

```
Transaction check succeeded.
```

```
Running transaction test
```

```
Transaction test succeeded.
```

```
Running transaction
Preparing      :                               1/1
Installing     : cowsay-3.04-10.fc28.noarch    1/1
Running scriptlet: cowsay-3.04-10.fc28.noarch  1/1
Verifying      : cowsay-3.04-10.fc28.noarch    1/1

Installed:
cowsay.noarch 3.04-10.fc28

Complete!
```

La commande `cowsay` fait exactement ce que son nom indique :

```
$ cowsay "Brought to you by yum"
```

```
< Brought to you by yum >
-----
      ^__^
      (oo)\_______
      (__)\       )\/\
          ||----w |
          ||     ||
```

Bien qu'elles puissent paraître inutiles, les commandes `figlet` et `cowsay` permettent d'attirer l'attention d'autres utilisateurs sur des informations pertinentes.

## Suppression des Paquets

Les commandes utilisées pour installer les paquets sont les mêmes utilisées pour les supprimer. Toutes les commandes acceptent le mot-clé `remove` pour désinstaller un paquet installé : `apt-get remove nom_du_paquet` ou `apt remove nom_du_paquet` pour les paquets DEB et `yum remove nom_du_paquet` ou `dnf remove nom_du_paquet` pour les paquets RPM. La commande `sudo` est également nécessaire pour effectuer la suppression. Par exemple, pour supprimer le paquet *figlet* précédemment installé d'une distribution basée sur DEB :

```
$ sudo apt-get remove figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  figlet
```

```
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 741 kB disk space will be freed.
Do you want to continue? [Y/n] Y
```

Après confirmation de l'opération, le paquet est supprimé du système :

```
(Reading database ... 115775 files and directories currently installed.)
Removing figlet (2.2.5-2) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Une procédure similaire est effectuée sur un système basé sur RPM. Par exemple, pour retirer le paquet *cowsay* précédemment installé d'une distribution basée sur RPM :

```
$ sudo yum remove cowsay
Dependencies resolved.
=====
Package            Arch             Version           Repository        Size
=====
Removing:
cowsay              noarch           3.04-10.fc28      @fedora           76 k

Transaction Summary
=====
Remove 1 Package

Freed space: 76 k
Is this ok [y/N]: y
```

De même, une confirmation est demandée et le paquet est supprimé du système :

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing           :                               1/1
  Erasing              : cowsay-3.04-10.fc28.noarch    1/1
  Running scriptlet: cowsay-3.04-10.fc28.noarch    1/1
  Verifying            : cowsay-3.04-10.fc28.noarch    1/1

Removed:
```

```
cowsay.noarch 3.04-10.fc28
```

Complete!

Les fichiers de configuration des paquets supprimés sont conservés sur le système, de sorte qu'ils peuvent être réutilisés si le paquet est réinstallé à l'avenir.

## Les Applications de Bureautique

Les applications de bureautique sont utilisées pour éditer des fichiers tels que des textes, des présentations, des feuilles de calcul et d'autres formats couramment utilisés dans un environnement de bureau. Ces applications sont généralement organisées en collections appelées *suites bureautiques*.

Pendant longtemps, la suite bureautique la plus utilisée sous Linux a été la suite *OpenOffice.org*. OpenOffice.org était une version open source de la suite *StarOffice* publiée par *Sun Microsystems*. Quelques années plus tard, Sun a été racheté par *Oracle Corporation*, qui a à son tour transféré le projet à la *Fondation Apache* et OpenOffice.org a été renommé *Apache OpenOffice*. Entre-temps, une autre suite bureautique basée sur le même code source a été publiée par la *Document Foundation*, qui l'a nommée *LibreOffice*.

Les deux projets ont les mêmes caractéristiques de base et sont compatibles avec les formats de documents de *Microsoft Office*. Toutefois, le format de document préféré est le *format Open document*, un format de fichier entièrement ouvert et normalisé par l'ISO. L'utilisation de fichiers ODF garantit que les documents peuvent être transférés entre les systèmes d'exploitation et les applications de différents fournisseurs, tels que Microsoft Office. Les principales applications proposées par OpenOffice/LibreOffice sont les suivantes

Writer: : Éditeur de texte Calc: : Tableurs Impress: : Présentations Draw: : Dessin vectoriel Maths: : Formules mathématiques Base: : Base de données

LibreOffice et Apache OpenOffice sont tous deux des logiciels libres, mais LibreOffice est sous licence LGPLv3 et Apache OpenOffice est sous licence Apache 2.0. La distinction entre les licences implique que LibreOffice peut incorporer des améliorations apportées par Apache OpenOffice, mais Apache OpenOffice ne peut pas incorporer des améliorations apportées par LibreOffice. Cela, et une communauté de développeurs plus active, sont la raison pour laquelle la plupart des distributions adoptent LibreOffice comme suite bureautique par défaut.

## Navigateurs Web

Pour la plupart des utilisateurs, l'objectif principal d'un ordinateur est de fournir un accès à

l'Internet. De nos jours, les pages web peuvent agir comme une application complète, avec l'avantage d'être accessibles de n'importe où, sans qu'il soit nécessaire d'installer un logiciel supplémentaire. Cela fait du navigateur web l'application la plus importante du système d'exploitation, du moins pour l'utilisateur moyen.

L'une des meilleures sources pour s'informer sur le développement du web est le MDN Web Docs, disponible à l'adresse <https://developer.mozilla.org/>. Maintenu par Mozilla, le site est rempli de tutoriels pour les débutants et de documents de référence sur la plupart des technologies web modernes.

Les principaux navigateurs web dans l'environnement Linux sont *Google Chrome* et *Mozilla Firefox*. Chrome est un navigateur web maintenu par Google mais est basé sur le navigateur open source *Chromium*, qui peut être installé en utilisant le gestionnaire de paquets de la distribution et est entièrement compatible avec Chrome. Maintenu par Mozilla, une organisation à but non lucratif, Firefox est un navigateur dont les origines sont liées à Netscape, le premier navigateur web populaire à adopter le modèle Open Source. La Fondation Mozilla est très impliquée dans le développement des standards ouverts qui sous-tendent le web moderne.

Mozilla développe également d'autres applications, comme le client de messagerie électronique *Thunderbird*. De nombreux utilisateurs choisissent d'utiliser un webmail plutôt qu'une application de courrier électronique dédiée, mais un client comme Thunderbird offre des fonctionnalités supplémentaires et s'intègre mieux aux autres applications de bureautique.

## Multimédia

Par rapport aux applications web disponibles, les applications de bureautique restent la meilleure option pour la création de contenu multimédia. Les activités liées au multimédia, comme le rendu vidéo, nécessitent souvent de grandes quantités de ressources système, qui sont mieux gérées par une application de bureautique locale. Certaines des applications multimédias les plus populaires pour l'environnement Linux et leurs utilisations sont énumérées ci-dessous.

### Blender

Un moteur de rendu 3D pour créer des animations. Blender peut également être utilisé pour exporter des objets 3D afin qu'ils soient imprimés par une imprimante 3D.

### GIMP

Un éditeur d'images complet, comparable à *Adobe Photoshop*, mais qui possède ses propres concepts et outils pour travailler avec les images. GIMP est utilisé pour créer, modifier et enregistrer la plupart des fichiers bitmap, comme les JPEG, PNG, GIF, TIFF et bien d'autres.

## Inkscape

Un éditeur de graphiques vectoriels, similaire à *Corel Draw* ou *Adobe Illustrator*. Le format par défaut d’Inkscape est SVG, qui est un standard ouvert pour les graphiques vectoriels. Les fichiers SVG peuvent être ouverts par n’importe quel navigateur web et, en raison de sa nature de graphique vectoriel, il peut être utilisé dans des mises en page flexibles de pages web.

## Audacity

Un éditeur audio. Audacity peut être utilisé pour filtrer, appliquer des effets et convertir entre de nombreux formats audios différents, comme MP3, WAV, OGG, FLAC, etc.

## ImageMagick

ImageMagick est un outil en ligne de commande permettant de convertir et de modifier la plupart des types de fichiers d’images. Il est également utilisé pour créer des documents PDF à partir de fichiers d’images et vice versa.

Il existe également de nombreuses applications dédiées à la lecture multimédia. L’application la plus populaire pour la lecture vidéo est *VLC*, mais certains utilisateurs préfèrent d’autres alternatives, comme *smplayer*. La lecture de musique en locale dispose également de nombreuses options, comme *Audacious*, *Banshee* et *Amarok*, qui peut également gérer une collection de fichiers audios locaux.

# Programmes Serveurs

Lorsqu’un navigateur web charge une page d’un site web, il se connecte en fait à un ordinateur distant et demande un élément d’information spécifique. Dans ce scénario, l’ordinateur qui exécute le navigateur web est appelé le *client* et l’ordinateur distant est appelé le *serveur*.

L’ordinateur serveur, qui peut être un ordinateur de bureau ordinaire ou du matériel spécialisé, a besoin d’un programme spécifique pour gérer chaque type d’information qu’il fournira. En ce qui concerne la desserte de pages web, la plupart des serveurs dans le monde déploient des programmes de serveur Open Source. Ce programme serveur particulier est appelé *serveur HTTP* (HTTP signifie *Hyper Text Transfer Protocol*) et les plus populaires sont *Apache*, *Nginx* et *lighttpd*.

Même de simples pages web peuvent nécessiter de nombreuses requêtes, qui peuvent être des fichiers ordinaires - appelés contenus statiques - ou des contenus dynamiques rendus à partir de diverses sources. Le rôle d’un serveur HTTP est de collecter et de renvoyer toutes les données demandées au navigateur, qui organise alors le contenu comme défini par le document HTML reçu (HTML signifie *Hyper Text Markup Language*) et d’autres fichiers d’appui. Par conséquent, le rendu d’une page web implique des opérations effectuées du côté serveur et des opérations effectuées du côté client. Les deux parties peuvent utiliser des scripts personnalisés pour accomplir des tâches spécifiques. Côté serveur HTTP, il est assez courant d’utiliser le langage de



script PHP. JavaScript est le langage de script utilisé du côté client (le navigateur web).

Les programmes de serveur peuvent fournir toutes sortes d'informations. Il est courant qu'un programme serveur demande des informations fournies par d'autres programmes serveurs. C'est le cas lorsqu'un serveur HTTP demande des informations fournies par un serveur de base de données.

Par exemple, lorsqu'une page dynamique est demandée, le serveur HTTP interroge généralement une base de données pour recueillir toutes les informations requises et renvoie le contenu dynamique au client. De la même manière, lorsqu'un utilisateur s'inscrit sur un site web, le serveur HTTP rassemble les données envoyées par le client et les stocke dans une base de données.

Une base de données est un ensemble organisé d'informations. Un serveur de base de données stocke le contenu de manière formatée, ce qui permet de lire, d'écrire et de relier de grandes quantités de données de manière fiable et rapide. Les serveurs de bases de données open source sont utilisés dans de nombreuses applications, et pas seulement sur internet. Même les applications locales peuvent stocker des données en se connectant à un serveur de base de données local. Le type de base de données le plus courant est la *base de données relationnelle*, dans laquelle les données sont organisées dans des tableaux prédéfinis. Les bases de données relationnelles open source les plus populaires sont *MariaDB* (issue de *MySQL*) et *PostgreSQL*.

## Partage des Données

Dans les réseaux locaux, comme ceux que l'on trouve dans les bureaux et les foyers, il est souhaitable que les ordinateurs puissent non seulement accéder à l'internet, mais aussi communiquer entre eux. Parfois, un ordinateur agit comme un serveur, parfois le même ordinateur agit comme un client. Cela est nécessaire lorsque l'on veut accéder à des fichiers sur un autre ordinateur dans le réseau. En occurrence, accéder à un fichier stocké sur un ordinateur de bureau à partir d'un ordinateur portable sans la difficulté de le copier sur une clé USB ou autre.

Entre les machines Linux, le *NFS (Network File System)* est souvent utilisé. Le protocole NFS est le moyen standard de partager des systèmes de fichiers dans des réseaux équipés uniquement de machines Unix/Linux. Avec NFS, un ordinateur peut partager un ou plusieurs de ses répertoires avec des ordinateurs spécifiques sur le réseau, afin qu'ils puissent lire et écrire des fichiers dans ces répertoires. NFS est même utilisé pour partager l'arborescence entière des répertoires d'un système d'exploitation avec des clients qui l'utiliseront pour démarrer. Ces ordinateurs, appelés clients légers, sont souvent utilisés dans les grands réseaux pour éviter la maintenance de chaque système d'exploitation sur chaque machine.

Si d'autres types de systèmes d'exploitation sont connectés au réseau, il est recommandé d'utiliser un protocole de partage de données qui peut être compris par tous. Cette exigence est remplie par *Samba*. Samba met en œuvre un protocole de partage de fichiers sur le réseau conçu à l'origine pour le système d'exploitation Windows, mais il est aujourd'hui compatible avec tous les principaux systèmes d'exploitation. Avec Samba, les ordinateurs du réseau local peuvent non seulement partager des fichiers, mais aussi des imprimantes.

Sur certains réseaux locaux, l'autorisation donnée lors de la connexion à un poste de travail est accordée par un serveur central, appelé *contrôleur de domaine*, qui gère l'accès aux différentes ressources locales et distantes. Le contrôleur de domaine est un service fourni par *Active Directory* de Microsoft. Les postes de travail Linux peuvent s'associer à un contrôleur de domaine en utilisant Samba ou un sous-système d'authentification appelé *SSSD*. À partir de la version 4, Samba peut également fonctionner comme contrôleur de domaine sur des réseaux hétérogènes.

Si l'objectif est de mettre en œuvre une solution de cloud computing capable de fournir diverses méthodes de partage de données sur le web, deux alternatives doivent être envisagées : *ownCloud* et *Nextcloud*. Les deux projets sont très similaires car le Nextcloud est un dérivé du ownCloud, ce qui est habituel parmi les projets open source. De telles déclinaisons sont généralement appelées *fork*. Les deux projets offrent les mêmes fonctionnalités de base : partage et synchronisation de fichiers, espaces de travail collaboratifs, calendrier, contacts et courrier, le tout via des interfaces de bureau, mobiles et web. Nextcloud propose également des conférences audio/vidéo privées, tandis que ownCloud est davantage axé sur le partage de fichiers et l'intégration avec des logiciels tiers. De nombreuses autres fonctionnalités sont fournies sous forme de plugins qui peuvent être activés ultérieurement si nécessaire.

ownCloud et Nextcloud offrent tous deux une version payante avec des fonctionnalités supplémentaires et un support étendu. Ce qui les différencie des autres solutions commerciales est la possibilité d'installer gratuitement le Nextcloud ou ownCloud sur un serveur privé, ce qui évite de conserver des données sensibles sur un serveur inconnu. Comme tous les services dépendent de la communication HTTP et sont écrits en PHP, l'installation doit être effectuée sur un serveur web préalablement configuré, comme Apache. Si vous envisagez d'installer ownCloud ou Nextcloud sur votre propre serveur, veillez à activer également le HTTPS pour chiffrer toutes les connexions à votre cloud.

## Administration du Réseau

La communication entre ordinateurs n'est possible que si le réseau fonctionne correctement. Normalement, la configuration du réseau est effectuée par un ensemble de programmes fonctionnant sur le routeur, chargé de mettre en place et de vérifier la disponibilité du réseau. Pour ce faire, deux services de base du réseau sont utilisés : *DHCP* (*Dynamic Host Configuration Protocol*) et *DNS* (*Domain Name System*).

Le DHCP est responsable de l'attribution d'une adresse IP à l'hôte lorsqu'un câble réseau est connecté ou lorsque l'appareil entre dans un réseau sans fil. Lors de la connexion à Internet, le serveur DHCP du fournisseur d'accès fournit une adresse IP à l'appareil demandeur. Un serveur DHCP est très utile dans les réseaux locaux également, pour fournir automatiquement des adresses IP à tous les appareils connectés. Si le DHCP n'est pas configuré ou s'il ne fonctionne pas correctement, il faudra configurer manuellement l'adresse IP de chaque appareil connecté au réseau. Il n'est pas pratique de définir manuellement les adresses IP sur les grands réseaux ou même sur les petits réseaux, c'est pourquoi la plupart des routeurs de réseau sont livrés avec un serveur DHCP préconfiguré.

L'adresse IP est nécessaire pour communiquer avec un autre appareil sur un réseau IP, mais les noms de domaine comme `www.lpi.org` sont beaucoup plus susceptibles d'être mémorisés qu'un numéro IP comme `203.0.113.165`. Le nom de domaine seul ne suffit cependant pas pour établir la communication par le réseau. C'est pourquoi le nom de domaine doit être traduit en une adresse IP par un serveur DNS. L'adresse IP du serveur DNS est fournie par le serveur DHCP du FAI et est utilisée par tous les systèmes connectés pour traduire les noms de domaine en adresses IP.

Les paramètres DHCP et DNS peuvent être modifiés en entrant dans l'interface web fournie par le routeur. Par exemple, il est possible de limiter l'attribution d'une adresse IP aux seuls appareils connus ou d'associer une adresse IP fixe à des machines spécifiques. Il est également possible de modifier le serveur DNS par défaut fourni par le fournisseur d'accès Internet. Certains serveurs DNS tiers, comme ceux fournis par Google ou OpenDNS, peuvent parfois fournir des réponses plus rapides et des fonctionnalités supplémentaires.

## Langages de Programmation

Tous les programmes informatiques (programmes clients et serveurs, applications de bureautique et le système d'exploitation lui-même) sont réalisés à l'aide d'un ou plusieurs langages de programmation. Les programmes peuvent être un simple fichier ou un système complexe de centaines de fichiers, que le système d'exploitation traite comme une séquence d'instructions à interpréter et à exécuter par le processeur et d'autres périphériques.

Il existe de nombreux langages de programmation à des fins très différentes et les systèmes Linux en fournissent beaucoup. Comme les logiciels libres comprennent également les sources des programmes, les systèmes Linux offrent aux développeurs des conditions parfaites pour comprendre, modifier ou créer des logiciels en fonction de leurs propres besoins.

Chaque programme commence sous la forme d'un fichier texte, appelé *code source*. Ce code source est écrit dans un langage plus ou moins convivial qui décrit ce que fait le programme. Un processeur d'ordinateur ne peut pas exécuter directement ce code. Dans les *langages compilés*, le

code source est donc converti en un *fichier binaire* qui peut ensuite être exécuté par l'ordinateur. Un programme appelé *compilateur* est chargé d'effectuer la conversion du code source en forme exécutable. Comme le binaire compilé est spécifique à un type de processeur, le programme peut devoir être recompilé pour être exécuté sur un autre type d'ordinateur.

Dans les *langages interprétés*, le programme n'a pas besoin d'être compilé au préalable. Au lieu de cela, un *interpréteur* lit le code source et exécute ses instructions chaque fois que le programme est exécuté. Cela rend le développement plus facile et plus rapide, mais en même temps les programmes interprétés ont tendance à être plus lents que les programmes compilés.

Voici quelques-uns des langages de programmation les plus populaires :

### JavaScript

JavaScript est un langage de programmation utilisé principalement dans les pages web. À l'origine, les applications JavaScript étaient très simples, comme les routines de validation de formulaires. Aujourd'hui, JavaScript est considéré comme un langage de première classe et il est utilisé pour créer des applications très complexes non seulement sur le web, mais aussi sur les serveurs et les appareils mobiles.

### C

Le langage de programmation C est étroitement lié aux systèmes d'exploitation, en particulier Unix, mais il est utilisé pour écrire tout type de programme sur presque tout type d'appareil. Les grands avantages du C sont la flexibilité et la rapidité. Le même code source écrit en C peut être compilé pour fonctionner sur différentes plates-formes et systèmes d'exploitation, avec peu ou pas de modifications. Cependant, après avoir été compilé, le programme ne s'exécutera que dans le système visé.

### Java

L'avantage principal de Java est que les programmes écrits dans ce langage sont portables, ce qui signifie qu'un même programme peut être exécuté dans différents systèmes d'exploitation. Malgré son nom, Java n'est pas apparenté à JavaScript.

### Perl

Perl est un langage de programmation très utilisé pour traiter le contenu des fichiers textes. Il met fortement l'accent sur les expressions régulières, ce qui fait de Perl un langage adapté au filtrage et à l'analyse de textes.

### Shell

Le shell, en particulier le shell Bash, est un langage de programmation, mais aussi une interface interactive pour exécuter d'autres programmes. Les programmes shells, appelés scripts shells, peuvent automatiser des tâches complexes ou répétitives dans l'environnement de la ligne de

commande.

## Python

Python est un langage de programmation très populaire parmi les étudiants et les professionnels qui ne sont pas directement impliqués dans l'informatique. Tout en ayant des fonctionnalités avancées, Python est un bon moyen de commencer à apprendre la programmation grâce son approche facile à utiliser.

## PHP

Le PHP est surtout utilisé comme langage de script côté serveur pour générer du contenu pour le web. La plupart des pages HTML en ligne ne sont pas des fichiers statiques, mais du contenu dynamique généré par le serveur à partir de diverses sources, comme des bases de données. Les programmes PHP, souvent simplement appelé pages PHP ou scripts PHP sont souvent utilisés pour générer ce genre de contenu. Le terme LAMP vient de la combinaison d'un système d'exploitation Linux, d'un serveur HTTP Apache, d'une base de données MySQL (ou MariaDB) et du langage de programmation PHP. Les serveurs LAMP sont une solution très populaire pour faire fonctionner des serveurs web. Outre PHP, tous les langages de programmation décrits précédemment peuvent également être utilisés pour mettre en œuvre de telles applications.

C et Java sont des langages compilés. Afin d'être exécuté par le système, le code source écrit en C est converti en code machine binaire, tandis que le code source Java est converti en *code binaire* exécuté dans un environnement logiciel spécial appelé *Machine Virtuelle Java*. JavaScript, Perl, script Shell, Python et PHP sont tous des langages interprétés, qui sont également appelés *langages de script*.

## Exercices Guidés

1. Pour chacune des commandes suivantes, identifiez si elle est associée au *système de gestion des paquets Debian* ou au *système de gestion des paquets Red Hat* :

dpkg	
rpm	
apt-get	
yum	
dnf	

2. Quelle commande pourrait être utilisée pour installer Blender sur Ubuntu ? Après l'installation, comment le programme peut-il être exécuté ?

3. Quelle application de la suite LibreOffice peut être utilisée pour travailler avec des feuilles de calcul électroniques ?

4. Quel navigateur web open-source est utilisé comme base pour le développement de Google Chrome ?

5. SVG est un standard ouvert pour les graphiques vectoriels. Quelle est l'application la plus populaire pour l'édition de fichiers SVG dans les systèmes Linux ?

6. Pour chacun des formats de fichiers suivants, inscrivez le nom d'une application capable d'ouvrir et de modifier le fichier correspondant :

png	
doc	
xls	
ppt	
wav	

7. Quel logiciel permet le partage de fichiers entre les machines Linux et Windows sur le réseau

local ?

## Exercices d'Exploration

1. Vous savez que les fichiers de configuration sont conservés même si le paquet associé est supprimé du système. Comment supprimeriez-vous automatiquement le paquet nommé *cups* et ses fichiers de configuration d'un système basé sur DEB ?
2. Supposons que vous ayez de nombreux fichiers d'images TIFF à convertir en JPEG. Quel logiciel utiliseriez-vous pour convertir ces fichiers directement en ligne de commande ?
3. Quel logiciel installerez-vous pour ouvrir les documents Microsoft Word qui vous sont envoyés par un utilisateur de Windows ?
4. Chaque année, linuxquestions.org promeut une enquête sur les applications Linux les plus populaires. Visitez <https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/> et découvrez quelles applications de bureau sont les plus populaires parmi les utilisateurs expérimentés de Linux.



# Résumé

Dans cette leçon, vous avez appris :

- Les systèmes de gestion des paquets utilisés dans les distributions majeures de Linux
- Des applications open source qui peuvent éditer des formats de fichiers populaires
- Les programmes de serveur qui sous-tendent de nombreux importants services Internet et de réseaux locaux
- Les langages de programmation communs et leurs usages

# Réponses aux Exercices Guidés

1. Pour chacune des commandes suivantes, identifiez si elle est associée au *système de gestion des paquets Debian* ou au *système de gestion des paquets Red Hat* :

dpkg	Système de gestion des paquets Debian
rpm	Système de gestion des paquets Red Hat
apt-get	Système de gestion des paquets Debian
yum	Système de gestion des paquets Red Hat
dnf	Système de gestion des paquets Red Hat

2. Quelle commande pourrait être utilisée pour installer Blender sur Ubuntu ? Après l'installation, comment le programme peut-il être exécuté ?

La commande `apt-get install blender`. Le nom du paquet doit être indiqué en minuscules. Le programme peut être exécuté directement depuis le terminal avec la commande `blender` ou en le choisissant dans le menu des applications.

3. Quelle application de la suite LibreOffice peut être utilisée pour travailler avec des feuilles de calcul électroniques ?

Calc

4. Quel navigateur web open-source est utilisé comme base pour le développement de Google Chrome ?

Chromium

5. SVG est un standard ouvert pour les graphiques vectoriels. Quelle est l'application la plus populaire pour l'édition de fichiers SVG dans les systèmes Linux ?

Inkscape

6. Pour chacun des formats de fichiers suivants, inscrivez le nom d'une application capable d'ouvrir et de modifier le fichier correspondant :

png	Gimp
doc	LibreOffice Writer
xls	LibreOffice Calc

ppt	LibreOffice Impress
wav	Audacity

7. Quel logiciel permet le partage de fichiers entre les machines Linux et Windows sur le réseau local ?

Samba

## Réponses aux Exercices d'Exploration

1. Vous savez que les fichiers de configuration sont conservés même si le paquet associé est supprimé du système. Comment supprimeriez-vous automatiquement le paquet nommé cups et ses fichiers de configuration d'un système basé sur DEB ?

```
apt-get purge cups
```

2. Supposons que vous ayez de nombreux fichiers d'images TIFF à convertir en JPEG. Quel logiciel convertirait ces fichiers directement en ligne de commande ?

ImageMagick

3. Quel logiciel installerez-vous pour ouvrir les documents Microsoft Word qui vous sont envoyés par un utilisateur de Windows ?

LibreOffice ou OpenOffice

4. Chaque année, linuxquestions.org promeut une enquête sur les applications Linux les plus populaires. Visitez <https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/> et découvrez quelles applications de bureau sont les plus populaires parmi les utilisateurs expérimentés de Linux.

Navigateur : Firefox. Client de messagerie : Thunderbird. Lecteur multimédia : VLC. Éditeur de graphiques raster : GIMP.



## 1.3 Logiciel à code source ouvert et licences

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 1.3

### Valeur

1

### Domaines de connaissance les plus importants

- Philosophie des logiciels à code source ouvert
- Licences des logiciels à code source ouvert
- Free Software Foundation (FSF), Open Source Initiative (OSI)

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Copyleft, Permissive
- GPL, BSD, Creative Commons
- Free Software, Open Source Software, FOSS, FLOSS
- Modèles économiques des logiciels à code source ouvert



# 1.3 Leçon 1

Certification :	Linux Essentials
Version :	1.6
Thème :	1 La communauté Linux et une Carrière dans l'Open Source
Objectif :	1.3 Les logiciels Open Source et les Licences
Leçon :	1 sur 1

## Introduction

Si les termes *logiciel libre* et *logiciel open source* sont largement utilisés, il existe encore quelques idées fausses sur leur signification. En particulier, le concept de “liberté” doit être examiné de plus près. Commençons par la définition des deux termes.

## Définition des Logiciels Libres et Open Source

### Critères de Logiciel Libre

Tout d’abord, “libre” dans le contexte du logiciel libre n’a rien à voir avec “gratuit”, ou comme le dit succinctement le fondateur de la *Free Software Foundation* (FSF), Richard Stallman :

Pour comprendre le concept, il faut penser “libre” comme dans “liberté d’expression”, et non comme dans “bière gratuite”.

— Richard Stallman, Qu’est-ce que le logiciel libre ?

Que vous deviez ou non payer pour le logiciel, il existe quatre critères que vérifie un logiciel libre. Richard Stallman décrit ces critères comme “les quatre libertés essentielles”, dont il commence le décompte à partir de zéro :

- “La liberté d’exécuter le programme comme vous le souhaitez, pour n’importe quel objectif (liberté 0).”

Où, comment et dans quel but le logiciel est utilisé ne peut être ni prescrit ni restreint.

- “La liberté d’étudier le fonctionnement du programme, et de le modifier pour qu’il fasse vos calculs comme vous le souhaitez (liberté 1). L’accès au code source est une condition préalable à cette liberté”.

Chacun peut modifier le logiciel en fonction de ses idées et de ses besoins. Cela suppose que le *code source*, c’est-à-dire tous les fichiers qui constituent un logiciel, soit disponible sous une forme lisible par les programmeurs. Et, bien sûr, ce droit s’applique à un utilisateur unique qui peut vouloir ajouter une seule fonction, ainsi qu’aux sociétés de logiciels qui construisent des systèmes complexes tels que les systèmes d’exploitation pour smartphones ou les microprogrammes pour routeurs.

- “La liberté de redistribuer des copies pour pouvoir aider les autres (liberté 2).”

Cette liberté encourage explicitement chaque utilisateur à partager le logiciel avec d’autres. Il s’agit donc d’une question de distribution la plus large possible et donc de la plus grande communauté possible d’utilisateurs et de développeurs qui, sur la base de ces libertés, développent et améliorent le logiciel au profit de tous.

- “La liberté de distribuer des copies de vos versions modifiées à d’autres personnes (liberté 3). En faisant cela, vous pouvez donner à toute la communauté une chance de bénéficier de vos modifications. L’accès au code source est une condition préalable.”

Il ne s’agit pas seulement de la distribution de logiciels libres, mais aussi de la distribution de logiciels libres *modifiés*. Toute personne qui apporte des modifications à un logiciel libre a le droit de mettre ces modifications à la disposition des autres. S’ils le font, ils sont obligés de le faire aussi librement, c’est-à-dire qu’ils ne doivent pas restreindre les libertés originales lors de la distribution du logiciel, même s’ils l’ont modifié ou étendu. Par exemple, si un groupe de développeurs a des idées différentes sur la direction d’un logiciel spécifique que les auteurs originaux, il peut se séparer dans une nouvelle branche de développement (appelée *fork*) et le poursuivre comme un nouveau projet. Mais, bien entendu, toutes les obligations associées à ces libertés demeurent.

L’accent mis sur l’idée de liberté est également cohérent dans la mesure où tout mouvement de

liberté est dirigé *contre* quelque chose, à savoir un opposant qui supprime les libertés mentionnées, qui considère le logiciel comme une propriété et qui veut le garder sous clé. Contrairement au logiciel libre, ce type de logiciel est appelé *propriétaire*.

## Logiciels Open Source vs Logiciels libres

Pour beaucoup, les *logiciels libres* et les *logiciels open source* sont synonymes. L'abréviation FOSS pour *Free and Open Source Software*, fréquemment utilisée, souligne ce point commun. FLOSS pour *Free/Libre and Open Source Software* est un autre terme populaire, qui souligne sans équivoque l'idée de liberté même pour d'autres langues que l'anglais. Cependant, si l'on considère l'origine et le développement de ces deux termes, une différenciation vaut la peine.

Le terme de *logiciel libre* avec la définition des quatre libertés décrites remonte à Richard Stallman et au projet GNU qu'il a fondé en 1985 environ 10 ans avant l'émergence de Linux. Le nom "GNU is not Unix" (GNU n'est pas Unix) décrit l'intention avec un clin d'œil : GNU a commencé comme une initiative visant à développer une solution techniquement convaincante nommément le système d'exploitation Unix à partir de zéro, à le rendre disponible au grand public et à l'améliorer continuellement avec le grand public. Pour cela, l'ouverture du code source était simplement une nécessité technique et organisationnelle, mais dans son propre image, le mouvement du logiciel libre est toujours un mouvement *social* et *politique* certains disent aussi que c'est un mouvement idéologique.

Avec le succès de Linux, les possibilités de collaboration d'Internet et les milliers de projets et d'entreprises qui ont vu le jour dans ce nouveau cosmos du logiciel, l'aspect social est de plus en plus relégué au second plan. L'ouverture du code source lui-même est passée d'une exigence technique à une caractéristique déterminante : dès que le code source était visible, le logiciel était considéré comme "open source". Les motifs sociaux ont fait place à une approche plus pragmatique du développement de logiciels.

Le logiciel libre et le logiciel open source fonctionnent sur la même base, avec les mêmes méthodes et dans une communauté mondiale d'individus, de projets et d'entreprises. Mais comme ils viennent de différentes directions une sociale et une pragmatique-technique il y a parfois des conflits. Ces conflits surviennent lorsque les résultats du travail commun ne correspondent pas aux objectifs initiaux des deux mouvements. Cela se produit en particulier lorsque le logiciel révèle ses sources mais ne respecte pas les quatre libertés du logiciel libre en même temps, par exemple lorsqu'il y a des restrictions sur la divulgation, le changement ou les connexions avec d'autres composants du logiciel.

La *licence* sous laquelle le logiciel est disponible détermine les conditions auxquelles un logiciel est soumis en ce qui concerne l'utilisation, la distribution et la modification. Et comme les exigences et les motifs peuvent être très différents, d'innombrables licences différentes ont été



créées dans le domaine des logiciels libres. En raison de l'approche beaucoup plus fondamentale du mouvement du logiciel libre, il est fréquent qu'il ne reconnaisse pas de nombreuses licences de logiciels libres comme "libres" et les rejette donc. Inversement, ce n'est guère le cas de l'approche open source en raison de son approche beaucoup plus pragmatique.

Examinons brièvement le domaine très complexe des licences ci-dessous.

## Licences

Contrairement à un réfrigérateur ou à une voiture, un logiciel n'est pas un produit *physique*, mais un produit numérique. Ainsi, une entreprise ne peut pas réellement transférer la propriété d'un tel produit en le vendant et en modifiant la possession physique du produit mais, elle transfère les droits d'utilisation de ce produit, et l'utilisateur accepte contractuellement ces droits d'utilisation. Les droits d'utilisation qui sont inscrits et surtout ceux qui *ne* sont *pas* inscrits dans la licence du logiciel, et l'on comprend donc l'importance des règles qui y sont contenues.

Alors que les grands vendeurs de logiciels propriétaires, tels que Microsoft ou SAP, ont leurs propres licences qui sont précisément adaptées à leurs produits, les partisans des logiciels libres et open source se sont efforcés dès le départ de clarifier et de généraliser la validité de leurs licences, car après tout, chaque utilisateur doit les comprendre et, si nécessaire, les utiliser lui-même pour ses propres développements.

Toutefois, il ne faut pas se cacher que cet idéal de simplicité peut difficilement être atteint, car trop d'exigences spécifiques et des conceptions juridiques pas toujours compatibles au niveau international s'y opposent. Pour ne citer qu'un exemple : Les droits d'auteur allemands et américains sont fondamentalement différents. Selon le droit allemand, il y a une *personne* en tant qu'*auteur* (plus précisément : *Urheber*), dont l'œuvre est sa *propriété intellectuelle*. Si l'auteur peut accorder l'autorisation d'utiliser son œuvre, il ne peut ni la céder ni renoncer à sa qualité d'auteur. Cette dernière est étrangère au droit américain. Ici aussi, il y a un auteur (qui peut cependant être aussi une entreprise ou une institution), mais il ne dispose que de droits d'exploitation qu'il peut transférer en partie ou en totalité et ainsi se détacher complètement de son œuvre. Une licence valable au niveau international doit être interprétée dans le respect des différentes législations.

Les conséquences sont nombreuses et parfois très différentes des licences FOSS. Pire encore, les différentes versions d'une licence, ou un mélange de licences (au sein d'un projet, ou même lors de la connexion de plusieurs projets) peuvent être source de confusion, voire de litiges.

Les représentants du logiciel libre et les partisans du mouvement open source, clairement orienté vers l'économie, ont créé leurs propres organisations, qui sont aujourd'hui résolument responsables de la formulation des licences de logiciels selon leurs principes et soutiennent leurs

membres dans leur application.

## Copyleft

La *Free Software Foundation* (FSF), déjà mentionnée, a formulé la *GNU General Public License* (GPL) comme l'une des plus importantes licences pour les logiciels libres, qui est utilisée par de nombreux projets, par exemple le noyau Linux. En outre, elle a publié des licences avec des adaptations spécifiques à chaque cas, telles que la *GNU Lesser General Public License* (LGPL, en français Licence Publique Générale Amointrie), qui régit la combinaison de logiciels libres avec des modifications apportées au code lorsque le code source des modifications ne doit pas être rendu public, la licence *GNU Affero General Public License* (AGPL, en français licence publique générale GNU Affero), qui couvre la vente d'accès à des logiciels hébergés, ou la licence *GNU Free Documentation License* (FDL, en français licence de documentation libre GNU), qui étend les principes de liberté à la documentation des logiciels. En outre, la FSF fait des recommandations pour ou contre les licences de tiers, et des projets affiliés tels que GPL-Violations.org enquêtent sur les violations présumées des licences libres.

La FSF évoque le principe selon lequel une licence libre s'applique également aux variantes modifiées du logiciel *copyleft* en contraste avec le principe du droit d'auteur restrictif qu'elle rejette. L'idée est donc de transférer les principes libéraux d'une licence de logiciel aussi librement que possible aux futures variantes du logiciel afin d'éviter des restrictions ultérieures.

Ce qui semble évident et simple entraîne cependant des complications considérables dans la pratique, raison pour laquelle les critiques qualifient souvent le principe du copyleft de "viral", puisqu'il est transmis aux versions ultérieures.

Il ressort de ce qui a été dit, par exemple, que deux composants logiciels qui sont sous licence de copyleft différente peuvent ne pas être combinables entre eux, puisque les deux licences ne peuvent pas être transférées au produit suivant en même temps. Cela peut même s'appliquer à différentes versions d'une même licence !

C'est pourquoi les nouvelles licences ou versions de licences ne saisissent souvent plus le copyleft avec autant de rigueur. La *GNU Lesser General Public License* (LGPL) déjà mentionnée est en ce sens une concession pour pouvoir connecter des logiciels libres avec des composants "non libres", comme cela se fait fréquemment avec les soi-disant *bibliothèques*. Les bibliothèques contiennent des sous-routines ou des routines, qui sont à leur tour utilisées par divers autres programmes. Cela conduit à la situation courante où un logiciel propriétaire appelle un tel sous-programme à partir d'une bibliothèque libre.

Une autre façon d'éviter les conflits de licence est la *double licence*, où un logiciel est licencié sous différentes licences, par exemple une licence libre et une licence propriétaire. Un cas d'utilisation

typique est celui d'une version libre d'un logiciel qui ne peut être utilisée que si elle respecte les restrictions de la copyleft et l'offre alternative d'obtenir le logiciel sous une licence différente qui libère le licencié de certaines restrictions en échange d'une redevance qui pourrait être utilisée pour financer le développement du logiciel.

Il devrait donc être clair que le choix de la licence pour les projets logiciels doit être fait avec beaucoup de prudence, car la coopération avec d'autres projets, la combinabilité avec d'autres composants et aussi la conception future du propre produit en dépendent. Le copyleft pose aux développeurs des défis particuliers à cet égard.

## Définition de l'Open Source et des Licences Permissives

Du côté de l'open source, c'est l'*Open Source Initiative* (OSI), fondée en 1998 par Eric S. Raymond et Bruce Perens, qui s'occupe principalement des questions de licence. Elle a également développé une procédure standardisée pour vérifier la conformité des licences de logiciels avec sa *définition de l'Open Source*. Plus de 80 licences Open Source reconnues sont actuellement disponibles sur le site de l'OSI.

Il y énumère également les licences "OSI-approved" qui contredisent explicitement le principe du copyleft, en particulier le groupe de *licences BSD*. La *Berkeley Software Distribution* (BSD) est une variante du système d'exploitation Unix développé à l'origine à l'université de Berkeley, qui a ensuite donné naissance à des projets libres tels que *NetBSD*, *FreeBSD* et *OpenBSD*. Les licences qui sous-tendent ces projets sont souvent qualifiées de *permissives*. Contrairement aux licences copyleft, elles n'ont pas pour but d'établir les conditions d'utilisation des variantes modifiées. La liberté maximale doit plutôt permettre une distribution aussi large que possible du logiciel en laissant les éditeurs du logiciel décider seuls de la manière de procéder avec les éditions quoi qu'il en soit, par exemple, ils les publient également ou les traitent comme des logiciels à code source fermé et les distribuent commercialement.

La *2-Clause BSD License* (licence BSD à 2 clauses), également appelée *Simplified BSD License* (licence BSD simplifiée) ou *FreeBSD License*, prouve à quel point une telle licence permissive peut être courte. En plus de la clause de responsabilité standardisée, qui protège les développeurs contre les actions en responsabilité découlant des dommages causés par le logiciel, la licence ne comporte que les deux règles suivantes :

La redistribution et l'utilisation sous forme source et binaire, avec ou sans modification, sont autorisées sous réserve que les conditions suivantes soient remplies :

1. Les redistributions de code source doivent conserver la notice de droit d'auteur ci-dessus, la présente liste de conditions et la clause de non-responsabilité suivante.
2. Les redistributions sous forme binaire doivent reproduire la notice de droit d'auteur ci-

dessus, la présente liste de conditions et l'avertissement suivant dans la documentation et/ou les autres documents fournis avec la distribution.

## Creative Commons

Le succès du concept de développement des FLOSS et les progrès technologiques associés ont conduit à des tentatives de transfert du principe de l'open source vers d'autres domaines non techniques. La préparation et la fourniture de connaissances, ainsi que la coopération créative dans la résolution de tâches complexes, sont désormais considérées comme des preuves du principe de l'open source étendu et lié au contenu.

Cela a également conduit à la nécessité de créer des bases fiables dans ces domaines, selon lesquelles les résultats des travaux peuvent être partagés et traités. Les licences de logiciels disponibles n'étant guère adaptées à cette fin, de nombreuses tentatives ont été faites pour convertir les exigences spécifiques du travail scientifique aux œuvres d'art numérisées “dans l'esprit de l'open source” en licences tout aussi pratiques.

L'initiative de loin la plus importante de ce type aujourd'hui est *Creative Commons* (CC), qui résume ses préoccupations comme suit :

Creative Commons est une organisation mondiale à but non lucratif qui permet le partage et la réutilisation de la créativité et des connaissances grâce à la mise à disposition d'outils juridiques gratuits.

— <https://creativecommons.org/faq/#what-is-creative-commons-and-what-do-you-do>

Avec Creative Commons, l'accent de la cession des droits revient du distributeur à l'auteur. Un exemple : Dans l'édition traditionnelle, un auteur cède généralement tous ses droits d'édition (impression, traduction, etc.) à un éditeur, qui assure à son tour la meilleure distribution possible de l'œuvre. Les canaux de distribution considérablement modifiés de l'internet mettent désormais l'auteur en mesure d'exercer lui-même un grand nombre de ces droits de publication et de décider lui-même de la manière dont son œuvre peut être utilisée. Creative Commons donne la possibilité de déterminer cela de manière simple et juridiquement fiable, mais Creative Commons veut plus : les auteurs sont encouragés à mettre leurs œuvres à disposition en tant que contribution à un processus général d'échange et de coopération. Contrairement au droit d'auteur traditionnel, qui donne à l'auteur tous les droits qu'il peut transférer à d'autres selon les besoins, l'approche de Creative Commons adopte l'approche inverse : l'auteur met son travail à la disposition de la communauté, mais peut choisir parmi un ensemble de fonctionnalités celles qui doivent être prises en compte lors de l'utilisation du travail — plus il choisit des fonctionnalités, plus la licence est restrictive.

Ainsi, le principe “Choose a License” de CC demande à l'auteur de préciser étape par étape les

propriétés individuelles et génère la licence recommandée, que l'auteur peut en dernier lieu attribuer à l'œuvre sous forme de texte et d'icône.

Pour une meilleure compréhension, voici un aperçu des six combinaisons et licences possibles offertes par CC :

### **CC BY ("Attribution")**

La licence libre qui permet à quiconque de modifier et de distribuer l'œuvre à condition de nommer l'auteur.

### **CC BY-SA ("Attribution-ShareAlike")**

Comme CC BY, mais l'œuvre modifiée ne peut être distribuée que sous la même licence. Le principe rappelle le copyleft, car la licence est "héritée" ici aussi.

### **CC BY-ND ("Attribution-NoDerivatives")**

Comme CC BY, mais le travail ne peut être transmis que sans modification.

### **CC BY-NC ("Attribution-NonCommercial")**

L'œuvre peut être éditée et distribuée en mentionnant le nom de l'auteur, mais uniquement dans des conditions non commerciales.

### **CC BY-NC-SA ("Attribution-NonCommercial-ShareAlike")**

Comme BY-NC, mais le travail ne peut être partagé que dans les mêmes conditions (c'est-à-dire une licence de type copyleft).

### **CC BY-NC-ND ("Attribution-NonCommercial-NoDerivatives")**

La licence la plus restrictive : la distribution est autorisée avec mention de l'auteur, mais uniquement sans modification et dans des conditions non commerciales.

## **Les Modèles Commerciaux dans l'Open Source**

Rétrospectivement, le triomphe des FLOSS agit comme un mouvement de base d'idéalistes technophiles qui, indépendamment des contraintes économiques et libres de toute dépendance monétaire, mettent leur travail au service du grand public. Dans le même temps, des entreprises valant des milliards ont été créées dans l'environnement des FLOSS ; pour n'en citer qu'une, la société américaine *Red Hat* fondée en 1993 avec un chiffre d'affaires annuel de plus de 3 milliards de dollars (2018), qui a été rachetée par le géant informatique IBM en 2018.

Examinons donc la tension entre la distribution gratuite et majoritairement gratuite de logiciels de haute qualité et les modèles commerciaux de leurs créateurs, car une chose doit être claire : les innombrables développeurs hautement qualifiés de logiciels libres doivent également gagner de

l'argent, et l'environnement FLOSS, à l'origine purement non commercial, doit donc développer des modèles commerciaux durables afin de préserver son propre cosmos.

Une approche commune, en particulier pour les grands projets dans la phase initiale, est le *crowdfunding*, c'est-à-dire la collecte de dons d'argent via une plateforme comme *Kickstarter*. En retour, les donateurs reçoivent une prime prédéfinie des développeurs en cas de succès, c'est-à-dire si des objectifs préalablement définis sont atteints, qu'il s'agisse d'un accès illimité au produit ou de caractéristiques spéciales.

Une autre approche est celle de la *double licence* : les logiciels libres sont proposés en parallèle sous une licence plus restrictive, voire propriétaire, ce qui garantit au client des services plus étendus (temps de réponse en cas d'erreurs, mises à jour, versions pour certaines plateformes, etc.) Un exemple parmi d'autres est *ownCloud*, qui est développé sous licence GPL et offre aux clients professionnels une "Edition Business" sous une licence propriétaire.

Prenons également *ownCloud* comme exemple d'un autre modèle commercial de FLOSS très répandu : les services professionnels. De nombreuses entreprises ne disposent pas des connaissances techniques internes nécessaires pour mettre en place et exploiter des logiciels complexes et critiques de manière fiable et, surtout, sûre. C'est pourquoi elles achètent des services professionnels tels que le conseil, la maintenance ou l'assistance technique directement auprès du fabricant. Les questions de responsabilité jouent également un rôle dans cette décision, car l'entreprise transfère les risques d'exploitation au fabricant.

Si un logiciel réussit à devenir performant et populaire dans son domaine, ce sont des possibilités périphériques de monétisation telles que le merchandising ou les certificats que les clients acquièrent et soulignent ainsi son statut particulier lors de l'utilisation de ce logiciel. La plateforme d'apprentissage *Moodle* propose la certification des formateurs, qui documentent leurs connaissances aux clients potentiels, par exemple, et ce n'est qu'un exemple parmi d'innombrables autres.

Le *Software as a Service* (SaaS) est un autre modèle commercial, en particulier pour les technologies basées sur le web. Dans ce cas, un fournisseur de services cloud exécute un logiciel comme un système de gestion de la relation client (CRM) ou un système de gestion de contenu (CMS) sur ses serveurs et permet à ses clients d'accéder à l'application installée. Cela permet au client d'éviter l'installation et la maintenance du logiciel. En retour, le client paie pour l'utilisation du logiciel en fonction de divers paramètres, par exemple le nombre d'utilisateurs. La disponibilité et la sécurité jouent un rôle important en tant que facteurs critiques pour l'entreprise.

Enfin, le modèle consistant à développer sur commande des extensions spécifiques au client pour en faire des logiciels libres est particulièrement courant dans les petits projets. Il appartient alors

généralement au client de décider de la manière de procéder avec ces extensions, c'est-à-dire s'il les publie également ou les garde sous clé dans le cadre de son propre modèle commercial.

Une chose devrait être claire : bien que les logiciels libres soient généralement disponibles gratuitement, de nombreux modèles commerciaux ont été créés dans leur environnement, qui sont constamment modifiés et étendus par d'innombrables freelances et entreprises dans le monde entier sous une forme très créative, ce qui assure finalement aussi la pérennité de l'ensemble du mouvement FLOSS.

# Exercices guidés

1. Quelles sont brièvement les "quatre libertés" telles que définies par Richard Stallman et la Free Software Foundation ?

liberté 0	
liberté 1	
liberté 2	
liberté 3	

2. Que signifie l'abréviation FLOSS ?

3. Vous avez développé un logiciel libre et vous voulez vous assurer que le logiciel lui-même, mais aussi tous les travaux futurs basés sur celui-ci, restent également libres. Quelle licence choisirez-vous ?

CC BY	
GPL version 3	
2-Clause BSD License	
LGPL	

4. Laquelle des licences suivantes qualifieriez-vous de permissive, laquelle qualifieriez-vous de copyleft ?

Simplified BSD License	
GPL version 3	
CC BY	
CC BY-SA	

5. Vous avez écrit une application web et l'avez publiée sous une licence libre. Comment pouvez-vous gagner de l'argent avec votre produit ? Citez trois possibilités.



## Exercices d'exploration

1. Sous quelle licence (y compris la version) les applications suivantes sont-elles disponibles ?

Apache HTTP Server	
MySQL Community Server	
Wikipedia articles	
Mozilla Firefox	
GIMP	

2. Vous souhaitez publier votre logiciel sous la licence GNU GPL v3. Quelles sont les étapes à suivre ?


3. Vous avez écrit un logiciel propriétaire et souhaitez le combiner avec un logiciel libre sous licence GPL version 3. Êtes-vous autorisé à le faire ou que devez-vous envisager ?


4. Pourquoi la Free Software Foundation a-t-elle publié la *GNU Affero General Public License* (GNU AGPL) en complément de la GNU GPL ?


5. Citez trois exemples de logiciels libres, qui sont également proposés en “Edition Business”, c’est-à-dire en version payante.


## Résumé

Dans cette leçon vous avez appris :

- Les similitudes et différences entre les logiciels libres et open source (FLOSS)
- Les licences FLOSS, leurs importances et leurs problèmes
- Copyleft vs licences permissives
- Les modèles commerciaux des FLOSS

## Réponses aux exercices guidés

1. Quelles sont – brièvement - les "quatre libertés" telles que définies par Richard Stallman et la Free Software Foundation ?

liberté 0	exécuter le logiciel
liberté 1	étudier et modifier le logiciel (code source)
liberté 2	distribuer le logiciel
liberté 3	distribuer le logiciel modifié

2. Que signifie l'abréviation FLOSS ?

Free/Libre Open Source Software

3. Vous avez développé un logiciel libre et vous voulez vous assurer que le logiciel lui-même, mais aussi tous les résultats futurs basés sur celui-ci, restent également libres. Quelle licence choisirez-vous ?

CC BY	
GPL version 3	X
2-Clause BSD License	
LGPL	

4. Laquelle des licences suivantes qualifieriez-vous de permissive, laquelle qualifieriez-vous de copyleft ?

Simplified BSD License	permissive
GPL version 3	copyleft
CC BY	permissive
CC BY-SA	copyleft

5. Vous avez écrit une application web et l'avez publiée sous une licence libre. Comment pouvez-vous gagner de l'argent avec votre produit ? Citez trois possibilités.

- Double licence, par exemple en offrant une "Edition Business" payante
- Offrir un hébergement, un service et un support
- Développer des extensions propriétaires pour les clients

# Réponses aux exercices d'exploration

1. Sous quelle licence (version incluse) les applications suivantes sont-elles disponibles ?

Apache HTTP Server	Apache License 2.0
MySQL Community Server	GPL 2
Wikipedia articles (English)	Creative Commons Attribution Share-Alike license (CC-BY-SA)
Mozilla Firefox	Mozilla Public License 2.0
GIMP	GPL 3

2. Vous souhaitez publier votre logiciel sous la licence GNU GPL v3. Quelles sont les étapes à suivre ?

- Si nécessaire, protégez-vous contre l'employeur en renonçant aux droits d'auteur, par exemple, afin de pouvoir préciser la licence.
- Ajoutez une notice de copyright à chaque fichier.
- Ajoutez à votre logiciel un fichier appelé COPYING avec le texte complet de la licence.
- Ajoutez une référence à la licence dans chaque fichier.

3. Vous avez écrit un logiciel propriétaire et souhaitez le combiner avec un logiciel libre sous licence GPL version 3. Êtes-vous autorisé à le faire ou que devez-vous envisager ?

Les FAQs de la Free Software Foundation fournissent des informations y relatives : À condition que votre logiciel propriétaire et le logiciel libre restent séparés l'un de l'autre, la combinaison est possible. Cependant, vous devez vous assurer que cette séparation est techniquement garantie et reconnaissable pour les utilisateurs. Si vous intégrez le logiciel libre de manière qu'il fasse partie de votre produit, vous devez également publier le produit sous licence GPL selon le principe du copyleft.

4. Pourquoi la Free Software Foundation a-t-elle publié la GNU Affero General Public License (GNU AGPL) en complément de la GNU GPL ?

La GNU AGPL comble une lacune de la licence qui se présente surtout avec les logiciels libres hébergés sur un serveur : Si un développeur apporte des modifications au logiciel, il n'est pas obligé, en vertu de la GPL, de rendre ces modifications accessibles, puisqu'il autorise l'accès au programme, mais ne "redistribue" pas sur le programme au sens de la GPL. La GNU AGPL, en revanche, stipule que le logiciel doit être mis à disposition pour le téléchargement avec toutes les modifications.

5. Citez trois exemples de logiciels libres, qui sont également proposés en “Edition Business”, par exemple dans une version payante.

MySQL, Zammad, Nextcloud



## 1.4 Compétences informatiques et travail sous Linux

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 1.4

### Valeur

2

### Domaines de connaissance les plus importants

- Compétences en bureautique
- Accéder à la ligne de commande
- Utilisations de Linux, de l'informatique en nuage et de la virtualisation dans l'industrie

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Utilisation d'un navigateur Web, préoccupations de confidentialité, options de configuration, recherche sur le Web et sauvegarde de contenu
- Terminal et console
- Problématiques de mots de passe
- Problématiques et outils de confidentialité
- Utilisation d'applications à code source ouvert courantes pour des présentations et des projets



## 1.4 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	1 La Communauté Linux et une Carrière dans l'Open Source
<b>Objectif :</b>	1.4 Compétences en TIC et Travail sous Linux
<b>Leçon :</b>	1 sur 1

### Introduction

Il fut un temps où travailler avec Linux sur un ordinateur de bureau était considéré comme difficile, car le système manquait de nombreuses applications de bureau et d'outils de configuration plus perfectionnés que ceux des autres systèmes d'exploitation. Certaines des raisons à cela étaient que Linux était beaucoup plus jeune que beaucoup d'autres systèmes d'exploitation. Cela étant dit, il était plus facile de commencer par développer des applications en ligne de commande plus essentielles et de laisser les outils graphiques plus complexes pour plus tard. Au début, puisque Linux était d'abord destiné à des utilisateurs plus avancés, cela n'aurait pas dû poser de problème. Mais cette époque est révolue depuis longtemps. Aujourd'hui, les environnements de bureau Linux sont très matures, ne laissant rien à désirer en ce qui concerne les fonctionnalités et la facilité d'utilisation. Néanmoins, la ligne de commande est toujours considérée comme un outil puissant utilisé chaque jour par les utilisateurs avancés. Dans cette leçon, nous examinerons quelques-unes des compétences de base en matière de bureautique dont vous aurez besoin pour choisir le meilleur outil pour le bon travail, y compris pour accéder à la ligne de commande.

## Interfaces Utilisateurs Linux

Lorsque vous utilisez un système Linux, vous interagissez soit avec une ligne de commande, soit avec une interface utilisateur graphique. Dans les deux cas, vous avez accès à de nombreuses applications qui vous permettent d'effectuer presque toutes les tâches avec l'ordinateur. Bien que l'objectif 1.2 vous ait déjà présenté une série d'applications couramment utilisées, nous commencerons cette leçon par un examen plus approfondi des environnements de bureau, des moyens d'accéder au terminal et des outils utilisés pour les présentations et la gestion de projets.

### Environnements de Bureau

Linux a une approche modulaire où différentes parties du système sont développées par différents projets et développeurs, chacun répondant à un besoin ou un objectif spécifique. De ce fait, il existe plusieurs options de bureau parmi lesquelles choisir, et avec les gestionnaires de paquets, l'environnement de bureau par défaut est l'une des principales différences entre les nombreuses distributions existantes. Contrairement aux systèmes d'exploitation propriétaires comme Windows et macOS, où les utilisateurs sont limités à l'environnement de bureau fourni avec leur système d'exploitation, il est possible d'installer plusieurs environnements et de choisir celui qui s'adapte le mieux à vous et à vos besoins.

Globalement, il existe deux principaux environnements de bureau dans le monde Linux : *Gnome* et *KDE*. Ils sont tous deux très complets, avec une grande communauté derrière eux et visent le même objectif, mais avec des approches légèrement divergentes. En bref, Gnome essaie de suivre le principe KISS ("keep it simple stupid"), avec des applications très rationalisées et propres. D'autre part, KDE a une autre perspective avec une plus grande sélection d'applications et en donnant à l'utilisateur la possibilité de changer chaque paramètre de configuration dans l'environnement.

Alors que les applications Gnome sont basées sur la boîte à outils GTK (écrite en langage C), les applications KDE utilisent la bibliothèque Qt (écrite en C++). L'un des aspects les plus pratiques de l'écriture d'applications avec la même boîte à outils graphique est que les applications auront tendance à partager une apparence et une convivialité similaires, qui est chargé de donner à l'utilisateur un sentiment d'unité au cours de son expérience. Une autre caractéristique importante est que le fait de partager la même bibliothèque graphique pour de nombreuses applications fréquemment utilisées peut permettre d'économiser de l'espace mémoire tout en accélérant le temps de chargement une fois que la bibliothèque a été chargée pour la première fois.

### Accéder à la Ligne de Commande

Pour nous, l'une des applications les plus importantes est l'émulateur de terminal graphique. On



les appelle des émulateurs de terminal parce qu'ils émulent réellement, dans un environnement graphique, les anciens terminaux série (souvent des machines Teletype) qui étaient en fait des clients connectés à une machine distante où les calculs s'effectuaient réellement. Ces machines étaient des ordinateurs très simples, sans aucun graphisme, qui étaient utilisés sur les premières versions d'Unix.

Dans Gnome, une telle application est appelée *Gnome Terminal*, tandis que dans KDE, elle se trouve sous le nom de *Konsole*. Mais il existe de nombreux autres choix, comme *Xterm*. Ces applications sont un moyen pour nous d'avoir accès à un environnement en ligne de commande afin de pouvoir interagir avec un shell.

Vous devez donc consulter le menu de la distribution de votre choix pour voir une application de terminal. Outre les différences entre elles, chaque application vous offrira ce qui est nécessaire pour utiliser la ligne de commande en toute confiance.

Une autre façon d'accéder au terminal est d'utiliser le TTY (TeleTYpewriter) virtuel. Vous pouvez y accéder en appuyant sur `Ctrl` + `Alt` + `F#`. Lisez `F#` comme l'une des touches de fonction de 1 à 7, par exemple. Il est probable que certaines des combinaisons initiales soient liées à l'utilisation de votre gestionnaire de session ou de votre environnement graphique. Les autres afficheront une invite vous demandant votre login comme celle qui suit :

```
Ubuntu 18.10 arrelia tty3
arrelia login:
```

`arrelia`, dans ce cas, est le nom d'hôte de la machine et `tty3` est le terminal disponible après avoir utilisé la combinaison de touches ci-dessus, plus la touche `F3`, comme dans `Ctrl` + `Alt` + `F3`.

Après avoir fourni votre login et votre mot de passe, vous pourrez enfin accéder à un terminal, mais sans environnement graphique, donc, vous ne pourrez pas utiliser la souris ou exécuter des applications graphiques sans d'abord démarrer une session X, ou Wayland. Mais cela dépasse le cadre de cette leçon.

## Présentations et Projets

L'outil le plus important pour les présentations sur Linux est *LibreOffice Impress*. Il fait partie de la suite bureautique open source appelée *LibreOffice*. Pensez à LibreOffice comme un remplacement open source de *Microsoft Office*. Il peut même ouvrir et enregistrer les fichiers PPT et PPTX qui sont natifs de *Powerpoint*. Mais malgré cela, je vous recommande vraiment d'utiliser le format natif ODP Impress. L'ODP fait partie de la famille *Open Document Format*, qui est une norme internationale pour ce type de fichier. C'est particulièrement important si vous voulez garder vos documents accessibles pendant de nombreuses années et moins vous soucier des

problèmes de compatibilité. Comme il s'agit d'un standard ouvert, il est possible pour n'importe qui de mettre en œuvre le format sans payer de droits d'auteur ou de licences. Cela vous permet également d'essayer librement d'autres logiciels de présentation de votre préférence et d'emporter vos fichiers avec vous, car il est très probable qu'ils seront compatibles avec ces nouveaux logiciels.

Mais si vous préférez le code aux interfaces graphiques, il existe quelques outils que vous pouvez choisir. *Beamer* est une classe *LaTeX* qui permet de créer des diaporamas à partir de code LaTeX. LaTeX lui-même est un système de composition largement utilisé pour la rédaction de documents scientifiques à l'académie, notamment pour sa capacité à traiter des symboles mathématiques complexes, que d'autres logiciels ont du mal à traiter. Si vous êtes à l'université et que vous devez traiter des équations et d'autres problèmes liés aux mathématiques, Beamer peut vous faire gagner beaucoup de temps.

L'autre option est *Reveal.js*, un super paquet NPM (NPM est le gestionnaire de paquets par défaut de NodeJS) qui vous permet de créer de belles présentations en utilisant le web. Ainsi, si vous pouvez écrire du HTML et du CSS, Reveal.js apportera beaucoup de fonctions JavaScript nécessaires pour créer de jolies présentations interactives qui s'adapteront bien à n'importe quelle résolution et taille d'écran.

Enfin, si vous souhaitez remplacer *Microsoft Project*, essayez *GanttProject* ou *ProjectLibre*. Les deux sont très similaires à leur homologue propriétaire et compatibles avec les fichiers Project.

## Utilisations de Linux par l'Industrie

Linux est très utilisé dans l'industrie du logiciel et de l'Internet. Des sites comme [W3Techs](https://www.w3techs.com/) indiquent qu'environ 68% des serveurs de sites web sur Internet sont alimentés par Unix et que la plus grande partie de ceux-ci sont sur Linux.

Cette large adoption est donnée non seulement pour la nature libre de Linux (comme la bière gratuite et la liberté d'expression) mais aussi pour sa stabilité, sa flexibilité et ses performances. Ces caractéristiques permettent aux fournisseurs de proposer leurs services à un coût réduit et avec une plus grande évolutivité. Une partie importante des systèmes Linux fonctionnent aujourd'hui dans le cloud, soit sur un modèle IaaS (Infrastructure as a service), PaaS (Platform as a Service) ou SaaS (Software as a Service).

L'IaaS est un moyen de partager les ressources d'un grand serveur en donnant accès à des machines virtuelles qui sont, en fait, plusieurs systèmes d'exploitation fonctionnant comme des invités sur une machine hôte par l'intermédiaire d'un important logiciel appelé *hyperviseur*. L'hyperviseur est chargé de permettre le fonctionnement de ces systèmes d'exploitation invités en séparant et en gérant les ressources disponibles sur la machine hôte. C'est ce que nous appelons la

*virtualisation*. Dans le modèle IaaS, vous ne payez que pour la fraction des ressources utilisées par votre infrastructure.

Linux dispose de trois hyperviseurs open source bien connus : *Xen*, *KVM* et *VirtualBox*. Xen est probablement le plus ancien d'entre eux. KVM a fait de Xen l'hyperviseur Linux le plus connu. Son développement est parrainé par RedHat, qui l'utilise, ainsi que d'autres acteurs, tant dans les services de cloud public que dans les installations de cloud privé. VirtualBox appartient à Oracle depuis son acquisition de Sun Microsystems et est généralement utilisé par les utilisateurs finaux en raison de sa facilité d'utilisation et d'administration.

Le PaaS et le SaaS, d'autre part, s'appuient sur le modèle IaaS, tant sur le plan technique que conceptuel. En PaaS, au lieu d'une machine virtuelle, les utilisateurs ont accès à une plateforme où il sera possible de déployer et d'exécuter leur application. L'objectif est ici d'alléger la charge de travail liée aux tâches d'administration du système et aux mises à jour des systèmes d'exploitation. Heroku est un exemple connu de PaaS où le code du programme peut être exécuté sans s'occuper des conteneurs et des machines virtuelles sous-jacents.

Enfin, le SaaS est le modèle où l'on paie généralement un abonnement pour pouvoir utiliser un logiciel sans se soucier de rien d'autre. *Dropbox* et *Salesforce* sont deux bons exemples de SaaS. La plupart de ces services sont accessibles via un navigateur web.

Un projet comme *OpenStack* est une collection de logiciels libres qui peuvent utiliser différents hyperviseurs et autres outils afin d'offrir un environnement IaaS complet en local (on premise), en exploitant la puissance d'un cluster d'ordinateurs sur votre propre datacenter. Toutefois, la mise en place d'une telle infrastructure n'est pas anodine.

## Problèmes de Protection de la Vie Privée lors de l'Utilisation d'Internet

De nos jours, le navigateur web est un élément fondamental de tout logiciel de bureau, mais certaines personnes n'ont pas encore les connaissances nécessaires pour l'utiliser en toute sécurité. Alors que de plus en plus de services sont accessibles via un navigateur web, presque toutes les actions effectuées par le biais d'un navigateur sont suivies et analysées par différentes parties. Sécuriser l'accès aux services internet et empêcher le suivi est un aspect important de l'utilisation sûre de l'internet.

### Suivi des cookies

Supposons que vous ayez navigué sur un site de commerce électronique, sélectionné un produit que vous vouliez et placé celui-ci dans le panier. Mais à la dernière seconde, vous avez décidé d'y réfléchir à deux fois et d'y réfléchir un peu plus longuement sur si vous en aviez vraiment besoin.

Au bout d'un certain temps, vous commencez à voir des publicités de ce même produit qui vous suivent sur le web. Lorsque vous cliquez sur les publicités, vous êtes immédiatement renvoyé à la page du produit de ce magasin. Il n'est pas rare que les produits que vous avez placés dans le panier y soient toujours, attendant que vous vous décidiez à les consulter. Vous êtes-vous déjà demandé comment ils font ? Comment ils vous montrent la bonne publicité sur une autre page web ? La réponse à ces questions s'appelle le *suivi des cookies* (cookie tracking).

Les cookies sont de petits fichiers qu'un site web peut enregistrer sur votre ordinateur afin de stocker et de récupérer des informations qui peuvent vous être utiles pour votre navigation. Ils sont utilisés depuis de nombreuses années et constituent l'un des plus anciens moyens de stockage de données du côté client. Un bon exemple de leur utilisation est celui des cartes d'identité uniques pour les achats. Ainsi, si vous revenez sur le même site web dans quelques jours, le magasin peut se souvenir des produits que vous avez placés dans votre panier lors de votre dernière visite et vous faire gagner du temps pour les retrouver.

C'est généralement correct, puisque le site web vous offre une fonction utile et ne partage aucune donnée avec des tiers. Mais qu'en est-il des publicités qui vous sont présentées lorsque vous surfez sur d'autres pages web ? C'est là qu'interviennent les réseaux publicitaires. Les réseaux publicitaires sont des entreprises qui proposent d'une part des publicités pour des sites de commerce électronique comme celui de notre exemple, et d'autre part la monétisation des sites web. Les créateurs de contenu comme les blogueurs, par exemple, peuvent mettre à disposition un espace pour ces réseaux publicitaires sur leur blog, en échange d'une commission liée aux ventes générées par cette publicité.

Mais comment savent-ils quel produit vous montrer ? Ils le font généralement en enregistrant également un cookie du réseau publicitaire au moment où vous avez visité ou recherché un certain produit sur le site web de commerce électronique. Ce faisant, le réseau est en mesure de récupérer les informations contenues dans ce cookie partout où le réseau diffuse des publicités, ce qui permet d'établir une corrélation avec les produits qui vous intéressent. C'est généralement l'un des moyens les plus courants de suivre quelqu'un sur l'internet. L'exemple que nous avons donné ci-dessus utilise un commerce électronique pour rendre les choses plus tangibles, mais les plateformes de médias sociaux font de même avec leurs boutons "J'aime" ou "Partager" et leur connexion sociale.

Un moyen de vous en débarrasser est de ne pas autoriser les sites web tiers à stocker des cookies sur votre navigateur. De cette façon, seul le site web que vous visitez peut stocker ses cookies. Mais sachez que certaines fonctionnalités "légitimes" peuvent ne pas fonctionner correctement si vous faites cela, car de nombreux sites dépendent aujourd'hui de services tiers pour fonctionner. Vous pouvez donc rechercher un gestionnaire de cookies dans le répertoire des modules complémentaires de votre navigateur afin d'avoir un contrôle précis des cookies qui sont stockés sur votre machine.

## Do Not Track (DNT)

Une autre idée fausse courante est liée à une certaine configuration de navigateur mieux connue sous le nom de DNT. Il s'agit d'un acronyme pour "Do Not Track" (ne pas suivre) et il peut être activé sur n'importe quel navigateur actuel. Tout comme pour le mode privé, il n'est pas difficile de trouver des personnes qui pensent qu'elles ne seront pas suivies si elles ont cette configuration. Malheureusement, ce n'est pas toujours vrai. Actuellement, DNT n'est qu'un moyen pour vous de dire aux sites web que vous visitez que vous ne voulez pas qu'ils vous pistent. Mais, en fait, ce sont eux qui décideront s'ils respecteront votre choix ou non. En d'autres termes, DNT est un moyen de vous soustraire au suivi des sites web, mais il n'y a aucune garantie quant au respect de ce choix.

Techniquement, cela se fait simplement en envoyant un champ supplémentaire sur l'en-tête du protocole de requête HTTP (DNT : 1) lors de la demande de données à un serveur web. Si vous souhaitez en savoir plus sur ce sujet, le site web <https://allaboutdnt.com> est un bon point de départ.

## Fenêtres "privées"

Vous avez peut-être remarqué les guillemets dans le titre ci-dessus. C'est parce que ces fenêtres ne sont pas aussi privées que la plupart des gens le pensent. Les noms peuvent varier, mais on peut les appeler "mode privé", "incognito" ou "anonyme", selon le navigateur que vous utilisez.

Dans Firefox, vous pouvez l'utiliser facilement en appuyant sur les touches **Ctrl** + **Maj** + **P**. Dans Chrome, il suffit d'appuyer sur les touches **Ctrl** + **Maj** + **N**. En fait, il ouvre une toute nouvelle session, qui ne partage généralement pas la configuration ou les données de votre profil standard. Lorsque vous fermez la fenêtre privée, le navigateur supprime automatiquement toutes les données générées par cette session, ne laissant aucune trace sur l'ordinateur utilisé. Cela signifie qu'aucune donnée personnelle, comme l'historique, les mots de passe ou les cookies, n'est stockée sur cet ordinateur.

Ainsi, beaucoup de personnes comprennent mal ce concept en croyant qu'elles peuvent naviguer anonymement sur l'internet, ce qui n'est pas tout à fait vrai. Une chose que fait le mode vie privée ou incognito est d'éviter ce que nous appelons le "cookie tracking". Lorsque vous visitez un site web, celui-ci peut stocker sur votre ordinateur un petit fichier qui peut contenir un identifiant permettant de vous suivre. À moins que vous ne configuriez votre navigateur pour qu'il n'accepte pas les cookies de tiers, les réseaux publicitaires ou d'autres entreprises peuvent stocker et récupérer cette identification et suivre effectivement votre navigation sur les sites web. Mais comme les cookies stockés lors d'une session en mode privé sont supprimés dès que vous fermez cette session, ces informations sont perdues à jamais.

En outre, les sites web et autres tiers sur Internet peuvent encore utiliser de nombreuses autres

techniques pour vous suivre. Ainsi, le mode privé vous apporte un certain niveau d'anonymat, mais il n'est totalement privé que sur l'ordinateur que vous utilisez. Si vous accédez à votre compte de messagerie ou à un site bancaire depuis un ordinateur public, comme dans un aéroport ou un hôtel, il faut absolument y accéder en utilisant le mode privé de votre navigateur. Dans d'autres situations, il peut y avoir des avantages, mais vous devez savoir exactement quels risques vous évitez et lesquels sont sans effet. Chaque fois que vous utilisez un ordinateur accessible au public, sachez que d'autres menaces de sécurité telles que les logiciels malveillants ou les enregistreurs de frappe peuvent exister. Soyez prudent lorsque vous saisissez des informations personnelles, y compris des noms d'utilisateur et des mots de passe, sur de tels ordinateurs ou lorsque vous téléchargez ou copiez des données confidentielles.

## Choisir le Bon Mot de Passe

L'une des situations les plus difficiles auxquelles tout utilisateur est confronté est le choix d'un mot de passe sécurisé pour les services qu'il utilise. Vous avez certainement déjà entendu dire que vous ne devez pas utiliser de combinaisons courantes comme `azerty`, `123456` ou `654321`, ni de chiffres facilement devinables comme votre date de naissance (ou celle d'un parent) ou votre code postal. La raison en est qu'il s'agit de combinaisons évidentes et qu'elles sont les plus fréquentes lors d'une tentative d'accès à votre compte par un intrus.

Il existe des techniques connues pour créer un mot de passe sûr. L'une des plus connues consiste à composer une phrase qui rappelle ce service et à choisir les premières lettres de chaque mot. Supposons que je veuille créer un bon mot de passe pour Facebook, par exemple. Dans ce cas, je pourrais inventer une phrase comme "Je serais heureux si j'avais 1000 amis comme Mike". Choisissez la première lettre de chaque mot et le mot de passe final serait `Jshsj a1000acM`. Cela donnerait un mot de passe de 12 caractères qui serait assez long pour être à la fois difficile à deviner et facile à retenir (pour autant que je puisse me souvenir de la phrase et de l'"algorithme" permettant de retrouver le mot de passe).

Les phrases sont généralement plus faciles à retenir que les mots de passe, mais même cette méthode a ses limites. Nous devons aujourd'hui créer des mots de passe pour de nombreux services et comme nous les utilisons avec des fréquences différentes, il sera finalement très difficile de se souvenir de toutes les phrases au moment où nous en aurons besoin. Alors que pouvons-nous faire ? Vous pouvez répondre que la chose la plus sage à faire dans ce cas est de créer quelques bons mots de passe et de les réutiliser sur des services similaires, n'est-ce pas ?

Malheureusement, ce n'est pas non plus une bonne idée. Vous avez probablement aussi entendu dire que vous ne devriez pas réutiliser le même mot de passe entre différents services. Le problème est qu'un service spécifique peut divulguer votre mot de passe (oui, cela arrive tout le temps) et toute personne qui y a accès essaiera d'utiliser la même combinaison de courrier électronique et de mot de passe sur d'autres services populaires sur Internet en espérant que vous



ayez fait exactement cela : recycler les mots de passe. Et devinez quoi ? Au cas où ils auraient raison, vous finirez par avoir un problème non seulement sur un seul service, mais sur plusieurs d'entre eux. Et croyez-moi, nous avons tendance à penser que cela ne nous arrivera pas jusqu'à ce qu'il soit trop tard.

Alors, que pouvons-nous faire pour nous protéger ? L'une des approches les plus sûres disponibles aujourd'hui est l'utilisation de ce que l'on appelle un *gestionnaire de mots de passe*. Les gestionnaires de mots de passe sont des logiciels qui stockent essentiellement tous vos mots de passe et noms d'utilisateur dans un format chiffré qui peut être déchiffré par un mot de passe maître. De cette façon, vous n'avez besoin de vous souvenir que d'un seul bon mot de passe puisque le gestionnaire gardera tous les autres en sécurité pour vous.

*KeePass* est l'un des gestionnaires de mots de passe open source les plus connus et les plus riches en fonctionnalités. Il stocke vos mots de passe dans un fichier chiffré au sein de votre système de fichiers. Le fait qu'il soit open source est un point important pour ce type de logiciel car il garantit qu'ils n'utiliseront pas vos données car tout développeur peut vérifier le code et savoir exactement comment il fonctionne. Cela apporte un niveau de transparence qu'il est impossible d'atteindre avec un code propriétaire. *KeePass* est disponible sur la plupart des systèmes d'exploitation, y compris Windows, Linux et macOS ; ainsi que pour les systèmes mobiles comme iOS et Android. Il comprend également un système de plugins qui permet d'étendre ses fonctionnalités bien au-delà des paramètres par défaut.

*Bitwarden* est une autre solution open source qui a une approche similaire, mais au lieu de stocker vos données dans un fichier, elle utilisera un serveur dans le cloud. De cette façon, il est plus facile de garder tous vos appareils synchronisés et vos mots de passe facilement accessibles par le web. *Bitwarden* est l'un des rares projets qui rendra non seulement les clients, mais aussi le serveur cloud disponible en tant que logiciel open source. Cela signifie que vous pouvez héberger votre propre version de *Bitwarden* et la rendre disponible à tout le monde, comme votre famille ou les employés de votre entreprise. Cela vous donnera une certaine flexibilité mais aussi un contrôle total sur la façon dont leurs mots de passe sont stockés et utilisés.

L'une des choses les plus importantes à garder à l'esprit lorsque vous utilisez un gestionnaire de mots de passe est la création d'un mot de passe aléatoire pour chaque service différent, puisque vous n'aurez pas besoin de les rappeler de toute façon. Il serait inutile d'utiliser un gestionnaire de mots de passe pour stocker des mots de passe recyclés ou facilement devinables. Ainsi, la plupart d'entre eux vous offriront un générateur de mots de passe aléatoires que vous pourrez utiliser pour créer les vôtres.

## Chiffrement

Chaque fois que des données sont transférées ou stockées, des précautions doivent être prises

pour s'assurer que des tiers ne peuvent pas y accéder. Les données transférées sur l'internet passent par une série de routeurs et de réseaux où des tiers peuvent accéder au trafic du réseau. De même, les données stockées sur des supports physiques peuvent être lues par toute personne qui entre en possession de ces supports. Pour éviter ce type d'accès, les informations confidentielles doivent être chiffrées avant de quitter un appareil informatique.

## TLS

*Transport Layer Security* (TLS) est un protocole qui permet d'assurer la sécurité des connexions réseau en utilisant la cryptographie. TLS est le successeur du protocole SSL (*Secure Sockets Layer*) qui a été déprécié en raison de graves lacunes. TLS a également évolué à plusieurs reprises afin de s'adapter et de devenir plus sûr, sa version actuelle est 1.3. Il peut assurer à la fois la confidentialité et l'authenticité en utilisant ce que l'on appelle la cryptographie symétrique et à clé publique. En disant cela, nous voulons dire qu'une fois utilisé, vous pouvez être sûr que personne ne pourra écouter ou modifier votre communication avec ce serveur pendant cette session.

La leçon la plus importante est de reconnaître qu'un site web est digne de confiance. Vous devez rechercher le symbole du "cadenas" dans la barre d'adresse du navigateur. Si vous le souhaitez, vous pouvez cliquer dessus pour inspecter le certificat qui joue un rôle important dans le protocole HTTPS.

Le TLS est ce qui est utilisé sur le protocole HTTPS (*HTTP over TLS*) afin de permettre par le web l'envoi de données sensibles (comme votre numéro de carte de crédit). Expliquer le fonctionnement de TLS va bien au-delà de l'objectif de cet article, mais vous pouvez trouver plus d'informations sur [Wikipedia](#) et sur le [wiki de Mozilla](#).

## Chiffrement de Fichiers et Courriels avec GnuPG

Il existe de nombreux outils pour sécuriser les courriels, mais l'un des plus importants est certainement *GnuPG*. GnuPG est l'acronyme de *GNU Privacy Guard* et c'est une implémentation open source d'*OpenPGP* qui est un standard international codifié dans la RFC 4880.

GnuPG peut être utilisé pour signer, chiffrer et déchiffrer des textes, des courriels, des fichiers, des répertoires et même des partitions entières de disque. Il fonctionne avec la cryptographie à clé publique et est largement disponible. En bref, GnuPG crée une paire de fichiers qui contiennent vos clés publiques et privées. Comme son nom l'indique, la clé publique peut être accessible à tous et la clé privée doit être gardée secrète. Les gens utiliseront votre clé publique pour chiffrer des données que seule votre clé privée pourra déchiffrer.

Vous pouvez également utiliser votre clé privée pour signer tout fichier ou courriel qui peut être validé avec la clé publique correspondante. Cette signalisation numérique fonctionne de manière



analogue à la signature du monde réel. Tant que vous êtes le seul à posséder votre clé privée, le destinataire peut être sûr que c'est vous qui l'avez créée. En utilisant la fonction de hachage cryptographique, GnuPG garantit également qu'aucune modification n'a été effectuée après la signature, car toute modification du contenu invaliderait la signature.

GnuPG est un outil très puissant et dans une certaine mesure également complexe. Vous pouvez trouver plus d'informations sur son [site web](#) et sur le [wiki d'Archlinux](#) (le wiki d'Archlinux est une très bonne source d'information, même si vous n'utilisez pas Archlinux).

## Chiffrement des Disques

Un bon moyen de sécuriser vos données est de chiffrer l'ensemble de votre disque ou de votre partition. Il existe de nombreux logiciels libres que vous pouvez utiliser pour atteindre cet objectif. Leur mode de fonctionnement et le niveau de chiffrement qu'ils offrent varient aussi considérablement. Il existe deux méthodes de base : le chiffrement par *empilement* et le chiffrement par *bloc* de périphériques.

Les solutions de systèmes de fichiers empilés sont mises en œuvre sur des systèmes de fichiers existants. Lorsqu'on utilise cette méthode, les fichiers et les répertoires sont chiffrés avant d'être stockés sur le système de fichiers et déchiffrés après leur lecture. Cela signifie que les fichiers sont stockés sur le système de fichiers hôte sous une forme chiffrée (ce qui signifie que leur contenu, et généralement aussi leurs noms de fichiers/dossiers, sont remplacés par des données d'apparence aléatoire), mais à part cela, ils existent toujours dans ce système de fichiers comme ils le feraient sans chiffrement, en tant que fichiers normaux, liens symboliques, liens durs, etc.

D'autre part, le chiffrement des périphériques à blocs se fait en dessous de la couche du système de fichiers, ce qui garantit que tout ce qui est écrit sur un périphérique à blocs est chiffré. Si vous regardez le bloc alors qu'il est hors ligne, il ressemblera à une grande section de données aléatoires et vous ne pourrez même pas dire quel type de système de fichiers est utilisé sans le déchiffrer au préalable. Cela signifie que vous ne pouvez pas savoir ce qu'est un fichier ou un répertoire, quelle est sa taille et quel type de données il contient, car les métadonnées, la structure des répertoires et les autorisations sont également chiffrées.

Les deux méthodes ont leurs propres avantages et inconvénients. Parmi toutes les options disponibles, vous devriez jeter un coup d'œil à *dm-crypt*, qui est la norme de facto pour le chiffrement par bloc pour les systèmes Linux, puisqu'il est natif dans le noyau. Il peut être utilisé avec l'extension *LUKS* (*Linux Unified Key Setup*), qui est une spécification mettant en œuvre une norme indépendante de la plate-forme pour une utilisation avec divers outils.

Si vous voulez essayer une méthode empilable, vous devriez jeter un coup d'œil à *EncFS*, qui est probablement la méthode la plus simple pour sécuriser les données sous Linux car elle ne

nécessite pas de privilèges root pour être mise en œuvre et elle peut fonctionner sur un système de fichiers existant sans modifications.

Enfin, si vous avez besoin d'accéder à des données sur différentes plateformes, consultez Veracrypt. Il succède à Truecrypt et permet la création de supports et de fichiers chiffrés, qui peuvent être utilisés aussi bien sous Linux que sous MacOS et Windows.

## Exercices Guidés

1. Vous devez utiliser une "fenêtre privée" dans votre navigateur si vous souhaitez :

Naviguer de manière totalement anonyme sur Internet	
Ne laisser aucune trace sur l'ordinateur que vous utilisez	
Activer le TLS afin d'éviter le suivi des cookies	
Utiliser le DNT	
Utiliser la cryptographie lors de la transmission de données	

2. Qu'est-ce qu'OpenStack ?

Un projet qui permet la création d'IaaS privés	
Un projet qui permet la création de PaaS privés	
Un projet qui permet la création de SaaS privés	
Un hyperviseur	
Un gestionnaire de mots de passe open source	

3. Lesquelles des options ci-dessous sont des logiciels de chiffrement de disque valables ?

RevealJS, EncFS et dm-crypt	
dm-crypt et KeePass	
EncFS et Bitwarden	
EncFS et dm-crypt	
TLS et dm-crypt	

4. Sélectionnez vrai ou faux pour le chiffrement par le dispositif dm-crypt :

Les fichiers sont chiffrés avant d'être écrits sur le disque	
--	--

L'ensemble du système de fichiers est un blob chiffré	
Seuls les fichiers et les répertoires sont chiffrés, pas les liens symboliques	
Ne nécessite pas d'accès root	
Est un dispositif de chiffrement par bloc	

5. Beamer est :

Un mécanisme de chiffrement	
Un hyperviseur	
Un logiciel de virtualisation	
Un composant d'OpenStack	
Un outil de présentation LaTeX	

## Exercices d'Exploration

1. La plupart des distributions sont livrées avec Firefox installé par défaut (si la vôtre ne l'est pas, vous devrez d'abord l'installer). Nous allons installer une extension de Firefox appelée *Lightbeam*. Vous pouvez le faire soit en appuyant sur `Ctrl` + `Maj` + `A` et en cherchant "Lightbeam" dans le champ de recherche qui s'affichera sur l'onglet ouvert, soit en visitant la page de l'extension avec Firefox et en cliquant sur le bouton "Installer" : <https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/>. Après avoir fait cela, démarrez l'extension en cliquant sur son icône et commencez à visiter certaines pages web sur d'autres onglets pour voir ce qui se passe.
2. Qu'est-ce qui est le plus important lorsqu'on utilise un gestionnaire de mots de passe ?


3. Utilisez votre navigateur web pour naviguer sur <https://haveibeenpwned.com/>. Découvrez l'objectif du site web et vérifiez si votre adresse électronique est concernée par des fuites de données.


## Résumé

Le terminal est un moyen puissant d'interagir avec le système et il existe de nombreux outils utiles et très matures à utiliser dans ce genre d'environnement. Vous pouvez accéder au terminal graphique en cherchant dans le menu de votre environnement de bureau ou en appuyant sur `Ctrl + Alt + F#`.

Linux est largement utilisé dans l'industrie technologique pour offrir des services IaaS, PaaS et SaaS. Il existe trois principaux hyperviseurs qui jouent un rôle important dans le soutien de ces services : Xen, KVM et Virtualbox.

Le navigateur web est un important logiciel informatique de nos jours, mais il est nécessaire de comprendre certaines choses pour l'utiliser en toute sécurité. Le DNT est juste un moyen de dire au site web que vous ne voulez pas être suivi, mais il n'y a aucune garantie que votre demande soit respectée. Les fenêtres privées ne sont privées que pour l'ordinateur que vous utilisez, pour cette raison cela vous permet d'échapper au suivi des cookies.

TLS est capable de chiffrer votre communication sur Internet, mais vous devez être capable de reconnaître quand il est utilisé. L'utilisation de mots de passe forts est également très importante pour assurer votre sécurité. La meilleure idée est donc de déléguer cette responsabilité à un gestionnaire de mots de passe et de permettre au logiciel de créer des mots de passe aléatoires pour chaque site auquel vous vous connectez.

Une autre façon de sécuriser vos communications est de signer et de chiffrer vos dossiers de fichiers et vos courriels avec GnuPG. dm-crypt et EncFS sont deux alternatives pour chiffrer des disques entiers ou des partitions qui utilisent respectivement les méthodes de chiffrement par blocs et par flux.

Enfin, LibreOffice Impress est une alternative open source à Microsoft Powerpoint très complète, mais il existe Beamer et RevealJS si vous préférez créer des présentations en utilisant du code plutôt que des interfaces graphiques. ProjectLibre et GanttProject peuvent être de bon choix si vous avez besoin d'un remplacement de Microsoft Project.

## Réponses aux Exercices Guidés

1. Vous devez utiliser une "fenêtre privée" dans votre navigateur si vous souhaitez :

Naviguer de manière totalement anonyme sur Internet	
Ne laisser aucune trace sur l'ordinateur que vous utilisez	X
Activer le TLS afin d'éviter le suivi des cookies	
Utiliser le DNT	
Utiliser la cryptographie lors de la transmission de données	

2. Qu'est-ce qu'OpenStack ?

Un projet qui permet la création d'IaaS privés	X
Un projet qui permet la création de PaaS privés	
Un projet qui permet la création de SaaS privés	
Un hyperviseur	
Un gestionnaire de mots de passe open source	

3. Lesquelles des options ci-dessous sont des logiciels de chiffrement de disque valables ?

RevealJS, EncFS et dm-crypt	
dm-crypt et KeePass	
EncFS et Bitwarden	
EncFS et dm-crypt	X
TLS et dm-crypt	

4. Sélectionnez vrai ou faux pour le chiffrement par le dispositif dm-crypt :

Les fichiers sont chiffrés avant d'être écrits sur le disque	V
--	---

L'ensemble du système de fichiers est un blob chiffré	V
Seuls les fichiers et les répertoires sont chiffrés, pas les liens symboliques	F
Ne nécessite pas d'accès root	F
Est un dispositif de chiffrement par bloc	V

5. Beamer est :

Un mécanisme de chiffrement	
Un hyperviseur	
Un logiciel de virtualisation	
Un composant d'OpenStack	
Un outil de présentation LaTeX	X



## Réponses aux Exercices d'Exploration

1. La plupart des distributions sont livrées avec Firefox installé par défaut (si la vôtre ne l'est pas, vous devrez d'abord l'installer). Nous allons installer une extension de Firefox appelée *Lightbeam*. Vous pouvez le faire soit en appuyant sur `Ctrl` + `Maj` + `A` et en cherchant "Lightbeam" dans le champ de recherche qui s'affichera sur l'onglet ouvert, soit en visitant la page de l'extension avec Firefox et en cliquant sur le bouton "Installer" : <https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/>. Après avoir fait cela, démarrez l'extension en cliquant sur son icône et commencez à visiter certaines pages web sur d'autres onglets pour voir ce qui se passe.

Vous vous souvenez de ces cookies dont nous avons parlé et qui peuvent partager vos données avec différents services lorsque vous visitez un site web ? C'est exactement ce que cette extension va vous montrer. Lightbeam est une expérience de Mozilla qui tente de révéler les sites principaux et sites tiers avec lesquels vous interagissez en visitant une seule URL. Ce contenu n'est généralement pas visible pour l'utilisateur moyen et il peut montrer que parfois un seul site web est capable d'interagir avec une douzaine de services ou plus.

2. Qu'est-ce qui est le plus important lorsqu'on utilise un gestionnaire de mots de passe ?

Lorsque vous utilisez un gestionnaire de mots de passe, la chose la plus importante à avoir à l'esprit est de mémoriser votre mot de passe principal et d'utiliser un mot de passe aléatoire unique pour chaque service différent.

3. Utilisez votre navigateur web pour naviguer sur <https://haveibeenpwned.com/>. Découvrez l'objectif du site web et vérifiez si votre adresse électronique est concernée par des fuites de données.

Le site web tient une base de données des informations de connexion dont les mots de passe ont été affectés par une fuite de mot de passe. Il permet de rechercher une adresse électronique et indique si cette adresse a été incluse dans une base de données publique d'informations d'identification volées. Il y a de fortes chances que votre adresse électronique soit également touchée par une fuite. Si c'est le cas, assurez-vous que vous avez mis à jour vos mots de passe récemment. Si vous n'utilisez pas déjà un gestionnaire de mots de passe, jetez un coup d'œil à ceux qui sont recommandés dans cette leçon.



**Linux  
Professional  
Institute**

## **Sujet 2 : Trouver son chemin sur un système Linux**



## 2.1 Bases sur la ligne de commande

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 2.1

### Valeur

3

### Domaines de connaissance les plus importants

- Interpréteur de commandes élémentaire
- Syntaxe en ligne de commande
- Variables
- Mise entre apostrophes ou guillemets

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Bash
- `echo`
- `history`
- Variable d'environnement `PATH`
- `export`
- `type`



Linux  
Professional  
Institute

## 2.1 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version:</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objectif :</b>	2.1 Principes de Base de la Ligne de Commande
<b>Leçon :</b>	1 sur 2

### Introduction

Les distributions Linux modernes disposent d'un large éventail d'interfaces graphiques, mais un administrateur devra toujours savoir comment travailler avec la ligne de commande, ou *shell* comme on l'appelle. Le shell est un programme qui permet une communication en mode texte entre le système d'exploitation et l'utilisateur. Il s'agit généralement d'un programme en mode texte qui lit les entrées de l'utilisateur et les interprète comme des commandes du système.

Il existe différents shells sur Linux, en voici quelques-uns :

- Bourne-again shell (Bash)
- C shell (csh ou tcsh, the enhanced csh)
- Korn shell (ksh)
- Z shell (zsh)

Sous Linux, le plus courant est le shell Bash. C'est également celui qui sera utilisé ici dans les exemples ou les exercices.

Lorsqu'on utilise un shell interactif, l'utilisateur entre des commandes à une invite de commande

(prompt). Pour chaque distribution Linux, l'invite par défaut peut être un peu différente, mais elle suit généralement cette structure :

```
username@hostname current_directory shell_type
```

Sur Ubuntu ou Debian GNU/Linux, l'invite pour un utilisateur régulier ressemblera probablement à ceci :

```
carol@mycomputer:~$
```

L'invite du super-utilisateur ressemblera à ceci :

```
root@mycomputer:~#
```

Sur CentOS ou Red Hat Linux, l'invite pour un utilisateur régulier ressemblera plutôt à ceci :

```
[dave@mycomputer ~]$
```

Et l'invite du super-utilisateur ressemblera à ceci :

```
[root@mycomputer ~]#
```

Expliquons chaque élément de la structure :

### **username**

Nom de l'utilisateur qui exécute le shell

### **hostname**

Nom de l'hôte sur lequel tourne le shell. Il existe également une commande `hostname`, avec laquelle vous pouvez afficher ou définir le nom d'hôte du système.

### **current\_directory**

Le répertoire dans lequel le shell se trouve actuellement. Un `~` signifie que le shell se trouve dans le répertoire personnel de l'utilisateur courant.

### **shell\_type**

`$` indique que le shell est exécuté par un utilisateur régulier.

# indique que le shell est exécuté par le super-utilisateur `root`.

Comme nous n'avons pas besoin de privilèges spéciaux, nous utiliserons une invite non privilégiée dans les exemples suivants. Par souci de concision, nous utiliserons simplement le `$` comme invite.

## Structure de la Ligne de Commande

La plupart des lignes de commandes suivent la même structure de base :

```
commande [option(s)/parametre(s)...] [argument(s)...]
```

Prenons la commande suivante comme exemple :

```
$ ls -l /home
```

Expliquons l'objectif de chaque composante :

### commande

Programme que l'utilisateur va exécuter - `ls` dans l'exemple ci-dessus.

### option(s)/parametre(s)

Un "interrupteur" qui modifie le comportement de la commande d'une manière ou d'une autre, comme `-l` dans l'exemple ci-dessus. Les options sont accessibles sous une forme courte et une forme longue. Par exemple, `-l` est identique à `--format=long`.

Il est possible de combiner plusieurs options, pour la forme courte les lettres peuvent généralement être collées. Par exemple, les commandes suivantes font toutes la même chose :

```
$ ls -al
$ ls -a -l
$ ls --all --format=long
```

### argument(s)

Les données supplémentaires requises par le programme, comme un nom de fichier ou un chemin d'accès, comme `/home` dans l'exemple ci-dessus.

La seule partie obligatoire de cette structure est la commande elle-même. En général, tous les autres éléments sont facultatifs, mais un programme peut exiger que certaines options,

paramètres ou arguments soient spécifiés.

**NOTE**

La plupart des commandes affichent un bref aperçu des commandes disponibles lorsqu'elles sont exécutées avec le paramètre `--help`. Nous apprendrons bientôt d'autres moyens d'en savoir plus sur les commandes Linux.

## Types de Comportement des commandes

Le shell supporte deux types de commandes :

### Interne

Ces commandes font partie du shell lui-même et ne sont pas des programmes séparés. Il existe environ 30 commandes de ce type. Leur but principal est d'exécuter des tâches à l'intérieur du shell (par exemple `cd`, `set`, `export`).

### Externe

Ces commandes résident dans des fichiers individuels. Ces fichiers sont généralement des programmes ou des scripts binaires. Lorsqu'une commande qui n'est pas intégrée à un shell est exécutée, le shell utilise la variable `PATH` pour rechercher un fichier exécutable portant le même nom que la commande. En plus des programmes qui sont installés avec le gestionnaire de paquets de la distribution, les utilisateurs peuvent également créer leurs propres commandes externes.

La commande `type` indique le type d'une commande spécifique :

```
$ type echo
echo is a shell builtin
$ type man
man is /usr/bin/man
```

## Insertion des guillemets (Quoting)

En tant qu'utilisateur de Linux, vous devrez créer ou manipuler des fichiers ou des variables de différentes manières. C'est facile lorsque vous travaillez avec des noms de fichiers courts et des valeurs uniques, mais cela devient plus compliqué lorsque, par exemple, des espaces, des caractères spéciaux et des variables sont impliqués. Les shells offrent une fonction appelée "quoting" qui encapsule ces données en utilisant différents types de guillemets (" ", ' '). Dans Bash, il existe trois types de guillemets :

- Guillemets doubles

- Guillemets simples
- Caractères d'échappement

Par exemple, les commandes suivantes n'agissent pas de la même manière en raison des types de guillemets :

```
$ TWOWORDS="two words"
$ touch $TWOWORDS
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 words
$ touch "$TWOWORDS"
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol    0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56  words
$ touch '$TWOWORDS'
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 15:00 '$TWOWORDS'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol    0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56  words
```

#### NOTE

La ligne avec `TWOWORDS=` est une variable Bash que nous avons créée nous-mêmes. Nous introduirons les variables plus tard. Ceci est juste destiné à vous montrer comment les guillemets affectent la sortie des variables.

## Guillemets doubles

Les guillemets doubles indiquent au shell de prendre le texte entre les guillemets ("...") comme des caractères normaux. Tous les caractères spéciaux perdent leur signification, sauf le `$` (signe de dollar), `\` (barre oblique inversée) et ``` (guillemet inverse). Cela signifie que les variables, la substitution de commandes et les fonctions arithmétiques peuvent toujours être utilisées.

Par exemple, la substitution de la variable `$USER` n'est pas affectée par les guillemets doubles :

```
$ echo Je suis $USER
Je suis tom
$ echo "Je suis $USER"
Je suis tom
```



Le caractère espace, en revanche, perd sa signification en tant que séparateur d'arguments :

```
$ touch nouveau fichier
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 fichier
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 nouveau
$ touch "nouveau fichier"
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:19 nouveau fichier
```

Comme vous pouvez le voir, dans le premier exemple, la commande `touch` crée deux fichiers individuels, la commande interprète les deux chaînes de caractères comme des arguments individuels. Dans le second exemple, la commande interprète les deux chaînes comme un seul argument, elle ne crée donc qu'un seul fichier. Il est toutefois préférable d'éviter le caractère espace dans les noms de fichiers. À la place, on peut utiliser un trait de soulignement (`_`) ou un point (`.`).

## Guillemets simples

Les guillemets simples n'ont pas les exceptions des guillemets doubles. Elles révoquent toute signification particulière de chaque caractère. Prenons l'un des premiers exemples ci-dessus :

```
$ echo Je suis $USER
Je suis tom
```

En appliquant les guillemets simples, vous obtenez un résultat différent :

```
$ echo 'Je suis $USER'
Je suis $USER
```

La commande affiche maintenant la chaîne exacte sans substituer la variable.

## Caractères d'échappement

Nous pouvons utiliser des *caractères d'échappement* pour supprimer les significations spéciales des caractères de Bash. Revenons à la variable d'environnement `$USER` :

```
$ echo $USER
carol
```

On voit que par défaut, le contenu de la variable est affiché dans le terminal. Cependant, si nous devons faire précéder le signe du dollar d'une barre oblique inversée (\), la signification particulière du signe du dollar serait alors annulée. Cela empêchera Bash d'étendre la valeur de la variable au nom d'utilisateur de la personne qui exécute la commande, mais interprétera le nom de la variable littéralement :

```
$ echo \USER
USER
```

Si vous vous souvenez bien, nous pouvons obtenir des résultats similaires en utilisant les guillemets simples, qui affichent le contenu littéral de ce qui se trouve entre les guillemets simples. Cependant, le caractère d'échappement fonctionne différemment en demandant à Bash d'ignorer toute signification particulière que le caractère qu'il précède pourrait posséder.

## Exercices Guidés

1. Séparez les lignes ci-dessous en composantes : commande, option(s)/paramètre(s) et argument(s) :

- Exemple: `cat -n /etc/passwd`

Commande :	<code>cat</code>
Option :	<code>-n</code>
Argument :	<code>/etc/passwd</code>

- `ls -l /etc`

Commande :	
Option :	
Argument :	

- `ls -l -a`

Commande :	
Option :	
Argument :	

- `cd /home/user`

Commande :	
Option :	
Argument :	

2. Trouvez les types de commandes suivantes :

Exemple:

<code>pwd</code>	Commande interne
<code>mv</code>	Commande Externe
<code>cd</code>	

cat	
exit	

3. Résolvez les commandes suivantes qui utilisent les guillemets :

Exemple:

echo "\$HOME est mon répertoire personnel"	echo /home/user est mon répertoire personnel
touch "\$USER"	
touch 'touch'	

## Exercices d'Exploration

1. Avec une seule commande et en utilisant l'expansion des accolades dans Bash (voir la page de manuel pour Bash), créez 5 fichiers numérotés de 1 à 5 avec le préfixe jeu (jeu1, jeu2, ...).

2. Supprimez les 5 fichiers que vous venez de créer avec une seule commande, en utilisant un caractère spécial différent (voir *Pathname Expansion* dans les pages de manuel Bash).

3. Existe-t-il d'autres moyens de faire interagir deux commandes entre elles ? Quels sont-ils ?

# Résumé

Dans cet atelier, vous avez appris :

- Les concepts du shell Linux
- Qu'est-ce que le shell Bash
- La structure de la ligne de commande
- Une introduction à l'insertion des guillemets

Commandes utilisées dans les exercices :

## **bash**

Le shell le plus populaire sur les ordinateurs Linux.

## **echo**

Affichage de texte sur le terminal.

## **ls**

Listage du contenu d'un répertoire.

## **type**

Montrer comment une commande spécifique est exécutée.

## **touch**

Créer un fichier vide ou mettre à jour la date de modification d'un fichier existant.

## **hostname**

Afficher ou modifier le nom d'hôte d'un système.

## Réponses aux Exercices Guidés

1. Séparez les lignes ci-dessous en composantes : commande, option(s)/paramètre(s) et argument(s) :

- Exemple: `cat -n /etc/passwd`

Commande :	<code>cat</code>
Option :	<code>-n</code>
Argument :	<code>/etc/passwd</code>

- `ls -l /etc`

Commande :	<code>ls</code>
Option :	<code>-l</code>
Argument :	<code>/etc</code>

- `ls -l -a`

Commande :	<code>ls</code>
Option :	<code>-l -a</code>
Argument :	

- `cd /home/user`

Commande :	<code>cd</code>
Option :	
Argument :	<code>/home/user</code>

2. Trouvez les types des commandes suivantes :

<code>cd</code>	Commande interne
<code>cat</code>	Commande externe
<code>exit</code>	Commande interne

3. Résolvez les commandes suivantes qui utilisent des guillemets :

<code>touch "\$USER"</code>	<code>tom</code>
<code>touch 'touch'</code>	Crée un fichier nommé touch



## Réponses aux Exercices d'Exploration

1. Avec une seule commande et en utilisant l'expansion des accolades dans Bash (voir la page de manuel pour Bash), créez 5 fichiers numérotés de 1 à 5 avec le préfixe `jeu` (`jeu1`, `jeu2`, ...).

Des plages peuvent être utilisées pour exprimer les nombres de 1 à 5 dans une même commande :

```
$ touch jeu{1..5}
$ ls
jeu1  jeu2  jeu3  jeu4  jeu5
```

2. Supprimez les 5 fichiers que vous venez de créer avec une seule commande, en utilisant un caractère spécial différent (voir *Pathname Expansion* dans les pages de manuel Bash).

Comme tous les fichiers commencent par `jeu` et se terminent par un seul caractère (un nombre de 1 à 5 dans ce cas), `?` peut être utilisé comme caractère spécial pour le dernier caractère du nom de fichier :

```
$ rm jeu?
```

3. Existe-t-il d'autres moyens de faire interagir deux commandes entre elles ? Quels sont-ils ?

Oui, une commande peut, par exemple, écrire des données dans un fichier qui est ensuite traité par une autre commande. Linux peut également collecter la sortie d'une commande et l'utiliser comme entrée pour une autre commande. C'est ce qu'on appelle *piping* et nous en apprendrons davantage à ce sujet dans une prochaine leçon.



Linux  
Professional  
Institute

## 2.1 Leçon 2

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objective:</b>	2.1 Principes de base de la Ligne de Commande
<b>Lesson:</b>	2 sur 2

### Introduction

Tous les shells gèrent un ensemble d'informations d'état tout au long des sessions shell. Ces informations d'exécution peuvent changer pendant la session et influencer le comportement du shell. Ces données sont également utilisées par les programmes pour déterminer certains aspects de la configuration du système. La plupart de ces données sont stockées dans ce que l'on appelle des variables, que nous allons aborder dans cette leçon.

### Les Variables

Les variables sont des éléments de stockage de données, tels que du texte ou des nombres. Une fois définie, la valeur d'une variable peut être consultée ultérieurement. Les variables ont un nom qui permet d'accéder à une variable spécifique, même lorsque le contenu de la variable change. Elles constituent un outil très courant dans la plupart des langages de programmation.

Dans la plupart des shells Linux, il existe deux types de variables :

#### Variables locales

Ces variables ne sont disponibles que pour le processus shell actuel. Si vous créez une variable

locale et que vous lancez ensuite un autre programme à partir de ce shell, la variable n'est plus accessible pour ce programme. Comme elles ne sont pas héritées par les sous-processus, ces variables sont appelées *variables locales*.

## Variables d'environnement

Ces variables sont disponibles à la fois dans une session shell spécifique et dans des sous-processus issus de cette session shell. Ces variables peuvent être utilisées pour transmettre des données de configuration aux commandes qui sont exécutées. Comme ces programmes peuvent accéder à ces variables, elles sont appelées *variables d'environnement*. La majorité des variables d'environnement sont en majuscules (par exemple `PATH`, `DATE`, `USER`). Un ensemble de variables d'environnement par défaut fournit, par exemple, des informations sur le répertoire personnel de l'utilisateur ou le type de terminal. Parfois, l'ensemble de toutes les variables d'environnement est appelé *environnement*.

### NOTE

Les variables ne sont pas persistantes. Lorsque le shell dans lequel elles ont été définies est fermé, toutes les variables et leur contenu sont perdus. La plupart des shells fournissent des fichiers de configuration qui contiennent des variables qui sont définies à chaque fois qu'un nouveau shell est lancé. Les variables qui doivent être définies de manière permanente doivent être ajoutées à l'un de ces fichiers de configuration.

## Manipuler des Variables

En tant qu'administrateur du système, vous devrez créer, modifier ou supprimer les variables locales et d'environnement.

### Travailler avec des Variables Locales

Vous pouvez définir une variable locale en utilisant l'opérateur `=` (égal). Une simple affectation permet de créer une variable locale :

```
$ greeting=hello
```

### NOTE

Ne mettez pas d'espace avant ou après l'opérateur `=`.

Vous pouvez afficher n'importe quelle variable en utilisant la commande `echo`. La commande affiche généralement le texte de sa section argument :

```
$ echo greeting
```

```
greeting
```

Pour accéder à la valeur de la variable, vous devrez utiliser `$` (signe dollar) devant le nom de la variable.

```
$ echo $greeting  
hello
```

Comme on peut le voir, la variable a été créée. Ouvrez maintenant un autre shell et essayez d'afficher le contenu de la variable créée.

```
$ echo $greeting
```

Rien n'est affiché. Cela montre que les variables n'existent toujours que dans un shell spécifique.

Pour vérifier que la variable est bien une variable locale, essayez d'engendrer un nouveau processus et vérifiez si ce processus peut accéder à la variable. Nous pouvons le faire en démarrant un autre shell et en laissant ce shell exécuter la commande `echo`. Comme le nouveau shell est lancé dans un nouveau processus, il n'héritera pas des variables locales de son processus parent :

```
$ echo $greeting world  
hello world  
$ bash -c 'echo $greeting world'  
world
```

**NOTE** | Veillez à utiliser des guillemets simples dans l'exemple ci-dessus.

Pour supprimer une variable, vous devez utiliser la commande `unset` :

```
$ echo $greeting  
hey  
$ unset greeting  
$ echo $greeting
```

**NOTE** | `unset` nécessite le nom de la variable comme argument. Vous ne pouvez donc pas ajouter `$` au nom, car cela résoudrait la variable et ferait passer la valeur de la variable à `unset` au lieu du nom de la variable.

## Travailler avec des Variables Globales

Pour mettre une variable à la disposition des sous-processus, il faut la transformer d'une variable locale en une variable d'environnement. Cela se fait par la commande `export`. Lorsqu'elle est invoquée avec le nom de la variable, cette variable est ajoutée à l'environnement du shell :

```
$ greeting=hello
$ export greeting
```

### NOTE

Encore une fois, assurez-vous de ne pas utiliser `$` lorsque vous exécutez la commande `export` car vous voulez passer le nom de la variable au lieu de son contenu.

Une façon plus facile de créer la variable d'environnement est de combiner les deux méthodes ci-dessus, en attribuant la valeur de la variable dans la partie argument de la commande.

```
$ export greeting=hey
```

Vérifions à nouveau si la variable est accessible aux sous-processus :

```
$ export greeting=hey
$ echo $greeting world
hey world
$ bash -c 'echo $greeting world'
hey world
```

Une autre façon d'utiliser les variables d'environnement est de les utiliser devant des commandes. Nous pouvons tester cela avec la variable d'environnement `TZ` qui contient le fuseau horaire. Cette variable est utilisée par la commande `date` pour déterminer le fuseau horaire à afficher :

```
$ TZ=EST date
Thu 31 Jan 10:07:35 EST 2019
$ TZ=GMT date
Thu 31 Jan 15:07:35 GMT 2019
```

Vous pouvez afficher toutes les variables d'environnement à l'aide de la commande `env`.

## La Variable PATH

La variable `PATH` est l'une des variables d'environnement les plus importantes dans un système Linux. Elle stocke une liste de répertoires, séparés par deux points, qui contiennent des programmes exécutables éligibles comme commandes du shell Linux.

```
$ echo $PATH
/home/user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Pour ajouter un nouveau répertoire à la variable, vous devrez utiliser le signe deux-points (:).

```
$ PATH=$PATH:new_directory
```

Voici un exemple :

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ PATH=$PATH:/home/user/bin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin
```

Comme vous le voyez, `$PATH` est utilisé dans la nouvelle valeur attribuée à `PATH`. Cette variable est résolue lors de l'exécution de la commande et permet de s'assurer que le contenu original de la variable est préservé. Bien entendu, vous pouvez également utiliser d'autres variables dans l'affectation :

```
$ mybin=/opt/bin
$ PATH=$PATH:$mybin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin:/opt/bin
```

La variable `PATH` doit être manipulée avec prudence, car elle est cruciale pour le travail en ligne de commande. Considérons la variable `PATH` suivante :

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Pour savoir comment le shell invoque une commande spécifique, la commande `which` peut être

exécutée avec le nom de la commande comme argument. On peut, par exemple, essayer de savoir où est stocké la commande `nano` :

```
$ which nano
/usr/bin/nano
```

Comme on peut le voir, l'exécutable de `nano` est situé dans le répertoire `/usr/bin`. Supprimons le répertoire de la variable et vérifions si la commande fonctionne toujours :

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
```

Regardons encore une fois la commande `nano` :

```
$ which nano
which: no nano in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games)
```

Comme on peut le voir, la commande n'est pas trouvée, donc pas exécutable. Le message d'erreur explique également la raison pour laquelle la commande n'a pas été trouvée et dans quels endroits elle a été recherchée.

Remettons les répertoires et exécutons à nouveau la commande.

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which nano
/usr/bin/nano
```

Maintenant, notre commande fonctionne à nouveau.

**TIP**

L'ordre des éléments dans `PATH` définit également l'ordre de recherche. Le premier exécutable correspondant trouvé en parcourant les chemins s'il est exécuté.

## Exercices Guidés

1. Créez une variable locale `number`.

2. Créez une variable d'environnement `ORDER`, en utilisant l'une des deux méthodes ci-dessus.

3. Affichez les noms des variables et leurs contenus.

4. Quelles sont les portées des variables créées précédemment ?



## Exercices d'Exploration

1. Créez une variable locale `nr_files` et affectez-lui le nombre de lignes trouvées dans le fichier `/etc/passwd`. Astuce : Examinez la commande `wc` et la substitution de commande et n'oubliez pas les guillemets.

2. Créez une variable d'environnement `ME`. Attribuez-lui la valeur de la variable `USER`.

3. Ajoutez la valeur de la variable `HOME` à `ME`, avec le délimiteur `:. Affichez le contenu de la variable ME.`

4. En utilisant l'exemple de date ci-dessus, créez une variable appelée `today` et attribuez-lui la date de l'un des fuseaux horaires.

5. Créez une autre variable appelée `today1` et attribuez-lui la date du système.

# Résumé

Dans cet atelier, vous avez appris :

- Les types de variables
- Comment créer des variables
- Comment manipuler les variables

Commandes utilisées dans les exercices :

## **env**

Affiche l'environnement actuel.

## **echo**

Affiche du texte.

## **export**

Met les variables locales à la disposition des sous-processus.

## **unset**

Supprime une variable.

## Réponses aux Exercices Guidés

1. Créez une variable locale `number`.

```
$ number=5
```

2. Créez une variable d'environnement `ORDER`, en utilisant l'une des deux méthodes ci-dessus.

```
$ export ORDER=desc
```

3. Afficher les noms des variables et leur contenu.

```
$ echo number
number
$ echo ORDER
ORDER
$ echo $number
5
$ echo $ORDER
desc
```

4. Quelles sont les portées des variables créées précédemment ?
  - La portée de la variable locale `number` est limitée au shell actuel.
  - La portée de la variable d'environnement `ORDER` est le shell actuel et tous les sous-shells générés par celui-ci.

## Réponses aux Exercices d'Exploration

1. Créez une variable locale `nr_files` et affectez-lui le nombre de lignes trouvées dans le fichier `/etc/passwd`. Astuce : Examinez la commande `wc` et la substitution de commande et n'oubliez pas les guillemets.

```
$ nr_files=`wc -l /etc/passwd`
```

2. Créez une variable d'environnement `ME`. Attribuez-lui la valeur de la variable `USER`.

```
$ export ME=$USER
```

3. Ajoutez la valeur de la variable `HOME` à `ME`, avec le délimiteur `:`. Affichez le contenu de la variable `ME`.

```
$ ME=$ME:$HOME
$ echo $ME
user:/home/user
```

4. En utilisant l'exemple de date ci-dessus, créez une variable appelée `today` et attribuez la date pour l'un des fuseaux horaires.

Les paragraphes suivants utilisent les fuseaux horaires GMT et EST à titre d'exemple, mais toute sélection de fuseau horaire est valable.

```
$ today=$(TZ=GMT date)
$ echo $today
Thu 31 Jan 15:07:35 GMT 2019
```

ou

```
$ today=$(TZ=EST date)
$ echo $today
Thu 31 Jan 10:07:35 EST 2019
```

5. Créez une autre variable appelée `today1` et attribuez-lui la date du système.

En supposant que vous êtes en GMT :

```
$ today1=$(date)
```

```
$ echo $today1
```

```
Thu 31 Jan 10:07:35 EST 2019
```



## 2.2 Utilisation de la ligne de commande pour obtenir de l'aide

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 2.2

### Valeur

2

### Domaines de connaissance les plus importants

- Pages de manuel
- Pages d'information

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `man`
- `info`
- `/usr/share/doc/`
- `locate`



## 2.2 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objectif :</b>	2.2 Utilisation de la Ligne de Commande pour Obtenir de l'Aide
<b>Leçon :</b>	1 sur 1

### Introduction

La ligne de commande est un outil très complexe. Chaque commande a ses propres options uniques, c'est pourquoi la documentation est essentielle lorsque l'on travaille avec un système Linux. Outre le répertoire `/usr/share/doc/`, qui contient la majeure partie de la documentation, divers autres outils fournissent des informations sur l'utilisation des commandes Linux. Ce chapitre se concentre sur les méthodes pour accéder à cette documentation, dans le but d'obtenir de l'aide.

Il existe une multitude de méthodes pour obtenir de l'aide dans la ligne de commande de Linux. `man`, `help` et `info` ne sont que quelques-unes d'entre elles. Pour Linux Essentials, nous nous concentrerons sur `man` et `info` car ce sont les outils les plus utilisés pour obtenir de l'aide.

Un autre sujet de ce chapitre sera la localisation des fichiers. Vous travaillerez principalement avec la commande `locate`.

# Obtenir de l'Aide sur la Ligne de Commande

## Aide Intégrée

Lorsqu'elles sont lancées avec le paramètre `--help`, la plupart des commandes affichent quelques brèves instructions sur leur utilisation. Bien que toutes les commandes ne fournissent pas ce commutateur, c'est tout de même une bonne première tentative pour en savoir plus sur les paramètres d'une commande. Sachez que les instructions de `--help` sont souvent assez brèves par rapport aux autres sources de documentation dont nous parlerons dans la suite de cette leçon.

## Les Pages Man

La plupart des commandes fournissent une page de manuel ou une page "man". Cette documentation est généralement installée avec le logiciel et peut être consultée avec la commande `man`. La commande dont la page de manuel doit être affichée est ajoutée à `man` en tant qu'argument :

```
$ man mkdir
```

Cette commande ouvre la page de manuel de `mkdir`. Vous pouvez utiliser les touches flèches haut et bas ou la barre d'espace pour naviguer dans la page de manuel. Pour quitter la page de manuel, appuyez sur `q`.

Chaque page de manuel est divisée en un maximum de 11 sections, bien que beaucoup de ces sections soient facultatives :

Section	Description
NAME	Nom de la commande et brève description
SYNOPSIS	Description de la syntaxe de la commande
DESCRIPTION	Description des effets de la commande
OPTIONS	Options disponibles
ARGUMENTS	Arguments disponibles
FILES	Fichiers auxiliaires
EXAMPLES	Exemples d'utilisation de la ligne de commande
SEE ALSO	Références croisées aux sujets connexes



Section	Description
DIAGNOSTICS	Messages d'Avertissement (Warning) et d'Erreur (Error)
COPYRIGHT	Auteur(s) de la commande
BUGS	Toutes les limitations connues de la commande

En pratique, la plupart des pages de manuel ne contiennent pas toutes ces parties.

Les pages de manuel sont organisées en huit catégories, numérotées de 1 à 8 :

Catégorie	Description
1	Commande utilisateur
2	Appels système
3	Fonctions de la bibliothèque C
4	Pilotes et fichiers de périphériques
5	Fichiers de configuration et formats de fichiers
6	Jeux
7	Divers
8	Commandes de l'administrateur système
9	Fonctions du noyau (non standard)

Chaque page de manuel appartient à une section précise. Cependant, plusieurs sections peuvent contenir des pages de manuel portant le même nom. Prenons l'exemple de la commande `passwd`. Cette commande peut être utilisée pour modifier le mot de passe d'un utilisateur. Puisque `passwd` est une commande utilisateur, sa page de manuel se trouve dans la section 1. En plus de la commande `passwd`, le fichier de base de données des mots de passe `/etc/passwd` possède également une page de manuel qui s'appelle `passwd`. Comme ce fichier est un fichier de configuration, il appartient à la section 5. Lorsqu'on se réfère à une page de manuel, la catégorie est souvent ajoutée au nom de la page de manuel, comme dans `passwd(1)` ou `passwd(5)` pour identifier la page de manuel respective.

Par défaut, `man passwd` affiche la première page de manuel disponible, dans ce cas `passwd(1)`. La catégorie de la page de manuel souhaitée peut être spécifiée dans une commande telle que `man 1 passwd` ou `man 5 passwd`.

Nous avons déjà discuté de la manière de naviguer dans une page de manuel et de revenir à la

ligne de commande. En interne, `man` utilise la commande `less` pour afficher le contenu de la page de manuel. `less` vous permet de rechercher du texte à l'intérieur d'une page de manuel. Pour rechercher le mot `linux`, vous pouvez simplement utiliser `/linux` pour une recherche en avant à partir du point où vous êtes sur la page, ou `?linux` pour lancer une recherche en arrière. Cette action met en évidence tous les résultats correspondants et déplace la page vers la première correspondance mise en évidence. Dans les deux cas, vous pouvez taper `N` pour passer au résultat suivant. Pour obtenir plus d'informations sur ces fonctions supplémentaires, appuyez sur `H` et un menu contenant toutes les informations s'affichera.

## Les Pages Info

Les pages d'information sont un autre outil qui vous aidera à travailler avec le système Linux. Les pages d'information sont généralement plus détaillées que les pages de manuel et sont formatées en hypertexte, comme les pages web sur Internet.

Les pages d'information peuvent être affichées de cette manière :

```
$ info mkdir
```

Pour chaque page d'information, `info` lit un fichier d'information qui est structuré en nœuds individuels au sein d'une arborescence. Chaque nœud contient un sujet simple et la commande `info` contient des hyperliens qui peuvent vous aider à passer de l'un à l'autre. Vous pouvez accéder au lien en appuyant sur la touche Entrée tout en plaçant le curseur sur l'un des astérisques en tête.

Tout comme `man`, l'outil `info` comporte également des commandes de navigation dans les pages. Vous pouvez en savoir plus sur ces commandes en appuyant sur `?` quand vous êtes sur une page `info`. Ces outils vous aideront à naviguer plus facilement sur la page et à comprendre comment accéder aux nœuds et vous déplacer dans l'arborescence des nœuds.

## Le Répertoire `/usr/share/doc/`

Comme mentionné précédemment, le répertoire `/usr/share/doc/` stocke la plupart des documents relatifs aux commandes utilisées par le système. Ce répertoire contient un dossier pour la plupart des paquets installés sur le système. Le nom de ce dossier est généralement le nom du paquet et parfois sa version. Ces dossiers comprennent un fichier `README` ou `readme.txt` qui contient la documentation de base du paquet. Outre le fichier `README`, le dossier peut également contenir d'autres fichiers de documentation, tels que le changelog qui comprend l'historique du programme en détail, ou des exemples de fichiers de configuration pour le paquet spécifique.

Les informations contenues dans le fichier README varient d'un paquet à l'autre. Tous les fichiers sont écrits en texte simple, ils peuvent donc être lus avec votre éditeur de texte préféré. Le nombre exact et les types de fichiers dépendent du paquet. Consultez certains des dossiers pour avoir un aperçu de leur contenu.

## Localisation des fichiers

### La Commande `locate`

Un système Linux est construit à partir de nombreux répertoires et fichiers. Linux dispose de nombreux outils pour localiser un fichier particulier au sein d'un système. Le plus rapide est la commande `locate`.

`locate` recherche dans une base de données et affiche ensuite chaque nom qui contient la chaîne de caractères donnée en argument:

```
$ locate note
/lib/udev/keymaps/zepto-znote
/usr/bin/zipnote
/usr/share/doc/initramfs-tools/maintainer-notes.html
/usr/share/man/man1/zipnote.1.gz
```

La commande `locate` permet d'utiliser des jokers et des expressions régulières, de sorte que la chaîne de caractères recherchée ne doit pas nécessairement être équivalente au nom complet du fichier souhaité. Vous en apprendrez plus sur les expressions régulières dans un chapitre ultérieur.

Par défaut, `locate` se comporte comme si le motif était entouré d'astérisques, donc `locate PATTERN` est la même chose que `locate *PATTERN*`. Cela vous permet de ne fournir que des sous-chaînes au lieu du nom de fichier exact. Vous pouvez modifier ce comportement avec les différentes options que vous trouverez expliquées dans la page de manuel de `locate`.

Parce que `locate` fait une lecture dans d'une base de données, il se peut que vous ne trouviez pas un fichier que vous avez récemment créé. La base de données est gérée par un programme appelé `updatedb`. Il est généralement exécuté périodiquement, mais si vous avez des privilèges de root et que vous avez besoin que la base de données soit mise à jour immédiatement, vous pouvez exécuter vous-même la commande `updatedb` à tout moment.

## La Commande find

`find` est un autre outil très populaire qui est utilisé pour la recherche de fichiers. Cette commande a une approche différente de celle de la commande `locate`. La commande `find` recherche récursivement dans une arborescence de répertoires, y compris ses sous-répertoires. `find` effectue une telle recherche à chaque appel, elle ne maintient pas de base de données comme `locate`. Tout comme `locate`, `find` supporte également les jokers et les expressions régulières.

`find` nécessite au moins le chemin où il doit chercher comme argument. En outre, des expressions peuvent être ajoutées pour fournir des critères de filtrage des fichiers à afficher. Un exemple est l'expression `-name`, qui recherche des fichiers avec un nom spécifique :

```
~$ cd Downloads
~/Downloads
$ find . -name thesis.pdf
./thesis.pdf
~/Downloads
$ find ~ -name thesis.pdf
/home/carol/Downloads/thesis.pdf
```

La première commande `find` recherche le fichier dans le répertoire courant `Downloads`, tandis que la seconde recherche le fichier dans le répertoire personnel de l'utilisateur.

La commande `find` est très complexe, c'est pourquoi elle ne sera pas couverte par l'examen Linux Essentials. Cependant, c'est un outil puissant qui est particulièrement pratique.

## Exercices Guidés

1. Utilisez la commande `man` pour savoir ce que fait chaque commande :

Commande	Description
<code>ls</code>	Afficher le contenu d'un répertoire.
<code>cat</code>	
<code>cut</code>	
<code>cd</code>	
<code>cp</code>	
<code>mv</code>	
<code>mkdir</code>	
<code>touch</code>	
<code>wc</code>	
<code>passwd</code>	
<code>rm</code>	
<code>rmdir</code>	
<code>more</code>	
<code>less</code>	
<code>whereis</code>	
<code>head</code>	
<code>tail</code>	
<code>sort</code>	
<code>tr</code>	
<code>chmod</code>	
<code>grep</code>	

2. Ouvrez la page info de `ls` et identifiez le MENU.

- Quelles options y avez-vous ?

- Trouvez l'option qui vous permet de trier la sortie par heure de modification.

3. Affichez le chemin d'accès aux 3 premiers fichiers README. Utilisez la commande `man` pour identifier la bonne option de `locate`.

4. Créez un fichier appelé `test` dans votre répertoire personnel. Trouvez son chemin absolu avec la commande `locate`.

5. L'avez-vous trouvé immédiatement ? Qu'avez-vous dû faire pour que `locate` puisse le trouver ?

6. Recherchez le fichier `test` que vous avez précédemment créé, à l'aide de la commande `find`. Quelle syntaxe avez-vous utilisée et quel est son chemin absolu ?

## Exercices d'Exploration

1. Il y a une commande dans le tableau ci-dessus qui n'a pas de page `man`. Laquelle est-ce et pourquoi pensez-vous que la commande n'a pas de page de manuel ?

2. En utilisant les commandes du tableau ci-dessus, créez l'arborescence de fichiers suivante. Les noms qui commencent par une majuscule sont des répertoires et ceux qui sont en minuscules sont des fichiers.

```
User
├── Documents
│   ├── Hello
│   │   ├── hey2
│   │   ├── helloa
│   │   └── ola5
│   └── World
│       └── earth9
├── Downloads
│   ├── Music
│   └── Songs
│       ├── collection1
│       └── collection2
├── Test
│   └── passa
└── test
```

3. Affichez à l'écran le répertoire de travail courant, y compris les sous-répertoires.

4. Recherchez dans l'arborescence tous les fichiers qui se terminent par un numéro.

5. Supprimez toute l'arborescence des répertoires avec une seule commande.

# Résumé

Dans cette leçon vous avez appris :

- Comment obtenir de l'aide
- Comment utiliser la commande `man`
- Comment naviguer dans une page `man`
- Les différentes sections d'une page `man`
- Comment utiliser la commande `info`
- Comment naviguer entre les différents nœuds
- Comment rechercher des fichiers dans le système

Commandes utilisées dans les exercices :

## **man**

Affiche une page de manuel.

## **info**

Affiche une page d'information.

## **locate**

Recherche dans la base de données de `locate` les fichiers portant un nom spécifique.

## **find**

Recherche dans le système de fichiers dont les noms correspondent à un ensemble de critères de sélection.

## **updatedb**

Met à jour la base de données de `locate`.



# Réponses aux Exercices Guidés

1. Utilisez la commande `man` pour savoir ce que fait chaque commande :

Commande	Description
<code>ls</code>	Affiche le contenu d'un répertoire.
<code>cat</code>	Concatène ou visualise des fichiers texte
<code>cut</code>	Supprime des sections d'un fichier texte
<code>cd</code>	Déplace dans un autre répertoire
<code>cp</code>	Copie d'un fichier
<code>mv</code>	Déplace un fichier ou un répertoire (il peut également être utilisé pour le renommer)
<code>mkdir</code>	Crée un nouveau répertoire
<code>touch</code>	Crée un fichier ou modifie la date et l'heure de la dernière modification d'un fichier existant
<code>wc</code>	Compte le nombre de mots, de lignes ou d'octets d'un fichier
<code>passwd</code>	Modifie le mot de passe d'un utilisateur
<code>rm</code>	Supprime un fichier
<code>rmdir</code>	Supprime un répertoire
<code>more</code>	Visualisation des fichiers texte un écran à la fois
<code>less</code>	Visualise de fichiers texte, permet de faire défiler une ligne ou une page à la fois
<code>whereis</code>	Affiche le chemin d'accès d'un programme spécifique et des fichiers manuels associés
<code>head</code>	Affiche les premières lignes d'un fichier
<code>tail</code>	Affiche les dernières lignes d'un fichier
<code>sort</code>	Ordonne un fichier par ordre numérique ou alphabétique
<code>tr</code>	Traduit ou supprime des caractères d'un fichier

Commande	Description
chmod	Modifie les autorisations d'un fichier
grep	Recherche dans un fichier

2. Ouvrez la page info de ls et identifiez le MENU.

- Quelles options y avez-vous ?
  - Quels sont les fichiers listés
  - Quelles sont les informations listées
  - Trier la sortie
  - Détails sur la version du tri
  - Formatage général de la sortie
  - Formatage de l'horodatage des fichiers
  - Formatage des noms de fichiers
- Trouvez l'option qui vous permet de trier la sortie par heure de modification.

`-t` ou `--sort=time`

3. Affichez le chemin d'accès aux 3 premiers fichiers README. Utilisez la commande `man` pour identifier la bonne option de `locate`.

```
$ locate -l 3 README
/etc/alternatives/README
/etc/init.d/README
/etc/rc0.d/README
```

4. Créez un fichier appelé `test` dans votre répertoire personnel. Trouvez son chemin absolu avec la commande `locate`.

```
$ touch test
$ locate test
/home/user/test
```

5. L'avez-vous trouvé immédiatement ? Qu'avez-vous dû faire pour que `locate` puisse le trouver ?

```
$ sudo updatedb
```

Le fichier est nouvellement créé, il n'y a donc aucune trace de celui-ci dans la base de données.

6. Recherchez le fichier test que vous avez précédemment créé, à l'aide de la commande `find`. Quelle syntaxe avez-vous utilisée et quel est son chemin absolu ?

```
$ find ~ -name test
```

ou

```
$ find . -name test  
/home/user/test
```

## Réponses aux Exercices d'Exploration

1. Il y a une commande dans le tableau ci-dessus qui n'a pas de page man. Laquelle est-ce et pourquoi pensez-vous que la commande n'a pas de page de manuel ?

La commande `cd`. Elle n'a pas de page de manuel car c'est une commande intégrée au shell.

2. En utilisant les commandes du tableau ci-dessus, créez l'arborescence de fichiers suivante. Les noms qui commencent par une majuscule sont des répertoires et ceux qui sont en minuscules sont des fichiers.

```
User
├── Documents
│   ├── Hello
│   │   ├── hey2
│   │   ├── helloa
│   │   └── ola5
│   └── World
│       └── earth9
├── Downloads
│   ├── Music
│   └── Songs
│       ├── collection1
│       └── collection2
├── Test
│   └── passa
└── test
```

La solution est une combinaison des commandes `mkdir` et `touch`.

3. Afficher à l'écran le répertoire de travail courant, y compris les sous-dossiers.

```
$ ls -R
```

4. Recherchez dans l'arborescence tous les fichiers qui se terminent par un numéro.

```
$ find ~ -name "[*0-9]"
$ locate "[*0-9]"
```

5. Supprimez toute l'arborescence des répertoires avec une seule commande.

```
$ rm -r Documents Downloads Test test
```



## 2.3 Utilisation des répertoires et liste des fichiers

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 2.3

### Valeur

2

### Domaines de connaissance les plus importants

- Fichiers, répertoires
- Fichiers et répertoires cachés
- Répertoires personnels
- Chemins d'accès absolus et relatifs

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Options courantes de `ls`
- Listes récursives de fichiers
- `cd`
- `.` et `..`
- `home` et `~`



## 2.3 Leçon 1

### Introduction

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objectif :</b>	2.3 Utilisation des Répertoires et Listage des Fichiers
<b>Leçon:</b>	1 sur 2

### Fichiers et Répertoires

Le système de fichiers Linux est similaire aux systèmes de fichiers d'autres systèmes d'exploitation car il contient des *fichiers* et des *répertoires*. Les fichiers contiennent des données telles que du texte lisible par l'homme, des programmes exécutables ou des données binaires utilisées par l'ordinateur. Les répertoires sont utilisés pour créer une organisation au sein du système de fichiers. Les répertoires peuvent contenir des fichiers et d'autres répertoires.

```
$ tree
```

```
Documents
```

```
├── Mission-Statement.txt
```

```
└── Reports
```

```
    └── report2018.txt
```

```
1 directory, 2 files
```

Dans cet exemple, `Documents` est un répertoire qui contient un fichier (`Mission-Statement.txt`) et un *sous-répertoire* (`Reports`). Le répertoire `Reports` contient à son tour un fichier appelé `report2018.txt`. Le répertoire `Documents` est dit être le *parent* du répertoire `Reports`.

Si la commande `tree` n'est pas disponible sur votre système, installez-la en utilisant le gestionnaire de paquets de votre distribution Linux. Reportez-vous à la leçon sur la gestion des paquets pour savoir comment faire.

## Noms de Fichiers et de Répertoires

Les noms de fichiers et de répertoires dans Linux peuvent contenir des minuscules et des majuscules, des chiffres, des espaces et des caractères spéciaux. Cependant, comme de nombreux caractères spéciaux ont une signification particulière dans le shell Linux, il est conseillé de ne pas utiliser d'espaces ou de caractères spéciaux lors de la dénomination des fichiers ou des répertoires. Les espaces, par exemple, nécessitent le *caractère d'échappement* `\` pour être saisis correctement :

```
$ cd Mission\ Statements
```

Aussi, en se référant au nom de fichier `report2018.txt`. Les noms de fichiers peuvent contenir un *suffixe* qui vient après le point (`.`). Contrairement à Windows, ce suffixe n'a pas de signification particulière dans Linux ; il est là pour la compréhension humaine. Dans notre exemple, `.txt` nous indique qu'il s'agit d'un fichier en texte clair, bien qu'il puisse techniquement contenir n'importe quel type de données.

## Naviguer dans le Système de Fichiers

### Obtenir la Localisation Actuelle

Comme les shells Linux tels que Bash sont basés sur du texte, il est important de se souvenir de votre emplacement actuel lorsque vous naviguez dans le système de fichiers. L'*invite de commande* fournit cette information :

```
user@hostname ~/Documents/Reports $
```



Notez que les informations telles que `user` et `hostname` seront traitées dans les prochaines sections. Grâce à l'invite, nous savons maintenant que notre emplacement actuel se trouve dans le répertoire `Reports`. De même, la commande `pwd` (*print working directory*) affichera le répertoire de travail :

```
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
```

La relation des répertoires est représentée par une barre oblique (`/`). Nous savons donc que `Reports` est un sous-répertoire de `Documents`, qui est un sous-répertoire de `user`, qui est situé dans un répertoire appelé `home`. `home` ne semble pas avoir de répertoire parent, mais ce n'est pas vrai du tout. Le parent de `home` est appelé `root` (racine), et est représenté par la première barre oblique (`/`). Nous parlerons du répertoire `root` dans une section ultérieure.

Notez que la sortie de la commande `pwd` diffère légèrement du chemin donné à l'invite de commande. Au lieu de `/home/user`, l'invite de commande contient un tilde (`~`). Le tilde est un caractère spécial qui représente le répertoire personnel de l'utilisateur. Ce point sera abordé plus en détail dans la prochaine leçon.

## Listage des Contenus d'un Répertoire

Le contenu du répertoire actuel est listé avec la commande `ls` :

```
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Notez que `ls` ne fournit aucune information sur le répertoire parent. De même, par défaut, `ls` n'affiche aucune information sur le contenu des sous-répertoires. `ls` ne peut que "voir" ce qui se trouve dans le répertoire courant.

## Changement du Répertoire Courant

La navigation dans Linux se fait principalement avec la commande `cd`. De l'anglais *change directory*. En utilisant la commande `pwd` d'avant, nous savons que notre répertoire actuel est `/home/user/Documents/Reports`. Nous pouvons changer notre répertoire actuel en entrant un nouveau chemin :

```
user@hostname ~ $ cd /home/user/Documents
user@hostname ~/Documents $ pwd
/home/user/Documents
```

```
user@hostname ~/Documents $ ls
Mission-Statement.txt Reports
```

Depuis notre nouvel emplacement, nous pouvons “voir” `Mission-Statement.txt` et notre sous-répertoire `Reports`, mais pas le contenu de notre sous-répertoire. Nous pouvons retourner dans `Reports` de cette manière :

```
user@hostname ~/Documents $ cd Reports
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Nous sommes maintenant de retour à la case départ.

## Chemins Absolus et Relatifs

La commande `pwd` affiche toujours un *chemin absolu*. Cela signifie que le chemin contient chaque niveau du chemin, du début du système de fichiers (`/`) au bout (`Reports`). Les chemins absolus commencent toujours par un `/`.

```
/
├── home
│   ├── user
│   │   ├── Documents
│   │   │   └── Reports
```

Le chemin absolu contient toutes les informations nécessaires pour accéder à `Reports` depuis n’importe quel endroit du système de fichiers. L’inconvénient est qu’il est fastidieux à taper.

Le deuxième exemple (`cd Reports`) était beaucoup plus facile à taper. Il s’agit d’un exemple de *chemin relatif*. Les chemins relatifs sont plus courts mais n’ont de sens que par rapport à votre position actuelle. Considérez cette analogie : Je vous rends visite à votre domicile. Vous me dites que votre ami habite à côté de chez vous. Je comprendrai cette situation parce qu’elle est relative à ma situation actuelle. Mais si vous me dites cela par téléphone, je ne pourrai pas trouver la maison de votre ami. Vous devrez me donner l’adresse complète de la rue.

## Chemins Relatifs Spéciaux

Le shell Linux nous donne des moyens de raccourcir nos chemins lors de la navigation. Pour

révéler les premiers chemins spéciaux, nous entrons dans la commande `ls` avec le drapeau `-a`. Ce drapeau modifie la commande `ls` de manière que *tous* les fichiers et répertoires soient listés, y compris les fichiers et répertoires cachés :

```
user@hostname ~/Documents/Reports $ ls -a
.
..
report2018.txt
```

**NOTE**

Vous pouvez vous référer à la page [man](#) de la commande `ls` comprendre ce que `-a` fait ici.

Cette commande a révélé deux résultats supplémentaires : Il s'agit de chemins spéciaux. Ils ne représentent pas de nouveaux fichiers ou répertoires, mais plutôt des répertoires que vous connaissez déjà :

.

Indique l'*emplacement actuel* (dans ce cas, `Reports`).

..

Indique le *répertoire parent* (dans ce cas, `Documents`).

Il n'est généralement pas nécessaire d'utiliser le chemin relatif spécial pour l'emplacement actuel. Il est plus facile et plus compréhensible de taper `report2018.txt` que de taper `./report2018.txt`. Mais `./report2018.txt` a des usages que vous apprendrez dans les sections suivantes. Pour l'instant, nous allons nous concentrer sur le chemin relatif du répertoire parent :

```
user@hostname ~/Documents/Reports $ cd ..
user@hostname ~/Documents $ pwd
/home/user/Documents
```

L'exemple de `cd` est beaucoup plus facile lorsqu'on utilise `..` au lieu du chemin absolu. De plus, nous pouvons combiner ce modèle pour remonter très rapidement dans l'arborescence des fichiers.

```
user@hostname ~/Documents $ cd ../../
$ pwd
/home
```

## Exercices Guidés

1. Pour chacun des chemins suivants, identifiez s'il est *absolu* ou *relatif*:

/home/user/Downloads	
../Reports	
/var	
docs	
/	

2. Observez la structure de fichiers suivante. Note : Les répertoires se terminent par une barre oblique (/) lorsque `tree` est invoqué avec l'option `-F`. Vous aurez besoin de privilèges élevés afin d'exécuter la commande `tree` sur le répertoire racine (/). Ce qui suit est un exemple de sortie et n'est pas indicatif d'une structure de répertoires complète. Utilisez-le pour répondre aux questions suivantes :

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directories, 5 files
```

Utilisez cette structure pour répondre aux questions suivantes.

Un utilisateur entre les commandes suivantes :

```
$ cd /etc/udev  
$ ls -a
```

Quel sera le résultat de la commande `ls -a` ?

3. Saisissez la commande la plus courte possible pour chacun des cas suivants :

- Votre localisation actuelle est la racine (/). Entrez la commande pour naviguer vers `lost+found` situé dans le répertoire `home` (exemple) :

```
$ cd home/lost+found
```

- Votre localisation actuelle est la racine (/). Entrez la commande pour naviguer vers le répertoire nommé `/etc/network/`.
- 

- Votre localisation actuelle est `/home/user/Documents/`. Naviguez vers le répertoire nommé `/etc/`.
- 

- Votre localisation actuelle est `/etc/systemd/system/`. Naviguez vers le répertoire nommé `/home/user/`.
- 

4. Considérez les commandes suivantes :

```
$ pwd  
/etc/udev/rules.d  
$ cd ../../systemd/user  
$ cd ..  
$ pwd
```

Quel est le résultat de la commande `pwd` finale ?

## Exercices d'Exploration

1. Supposons qu'un utilisateur ait entré les commandes suivantes :

```
$ mkdir "this is a test"
$ ls
this is a test
```

Quelle commande `cd` vous permettrait d'entrer dans ce répertoire ?

2. Essayez à nouveau, mais après avoir tapé `cd this`, appuyez sur la touche TAB. Qu'est-ce qui est maintenant affiché à l'invite ?

C'est un exemple d'*autocomplétion*, qui est un outil inestimable non seulement pour gagner du temps, mais aussi pour prévenir les erreurs d'orthographe.

3. Essayez de créer un répertoire dont le nom contient un caractère `\`. Affichez le nom du répertoire avec `ls` puis supprimez le répertoire.

## Résumé

Dans cette leçon, vous avez appris :

- Les bases du système de fichiers Linux
- La différence entre les répertoires *parents* et les *sous-répertoires*
- La différence entre les chemins de fichier *absolus* et les chemins de fichier *relatifs*
- Les chemins relatifs spéciaux `.` et `..`
- Naviguer dans le système de fichiers à l'aide de `cd`
- Indiquez votre position actuelle en utilisant `pwd`
- Listez *tous* les fichiers et répertoires en utilisant `ls -a`

Les commandes suivantes ont été abordées dans cette leçon :

### **cd**

Modifie le répertoire actuel.

### **pwd**

Affiche le chemin du répertoire de travail actuel.

### **ls**

Liste le contenu d'un répertoire et affiche les propriétés des fichiers.

### **mkdir**

Crée un nouveau répertoire.

### **tree**

Afficher une liste hiérarchique d'une arborescence de répertoires.

## Réponses aux Exercices Guidés

1. Pour chacun des chemins suivants, identifiez s'il est *absolu* ou *relatif* :

/home/user/Downloads	absolu
../Reports	relatif
/var	absolu
docs	relatif
/	absolu

2. Observez la structure de fichiers suivante. Note : Les répertoires se terminent par une barre oblique (/) lorsque `tree` est invoqué avec l'option `-F`. Vous aurez besoin de privilèges élevés afin d'exécuter la commande `tree` sur le répertoire racine (/). Ce qui suit est un exemple de sortie et n'est pas indicatif d'une structure de répertoires complète. Utilisez-le pour répondre aux questions suivantes :

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
├── home/
│   ├── lost+found/
│   └── user/
│       └── Documents/

12 directories, 5 files
```

Un utilisateur entre les commandes suivantes :



```
$ cd /etc/udev  
$ ls -a
```

Quel sera le résultat de la commande `ls -a` ?

```
. .. rules.d udev.conf
```

3. Saisissez la commande la plus courte possible pour chacun des cas suivants :

- Votre localisation actuelle est la racine (/). Entrez la commande pour naviguer vers `lost+found` dans le répertoire `home` (exemple) :

```
$ cd home/lost+found
```

- Votre localisation actuelle est la racine (/). Entrez la commande pour naviguer vers le répertoire nommé `/etc/network` :

```
$ cd etc/network
```

- Votre localisation actuelle est `/home/user/Documents/`. Naviguez vers le répertoire nommé `/etc/`.

```
$ cd /etc
```

- Votre localisation actuelle est `/etc/systemd/system/`. Naviguez vers le répertoire nommé `/home/user/`.

```
$ cd /home/user
```

4. Considérez les commandes suivantes :

```
$ pwd  
/etc/udev/rules.d  
$ cd ../../systemd/user  
$ cd ..  
$ pwd
```

Quel est le résultat de la commande `pwd` finale ?

```
/etc/systemd
```

# Réponses aux Exercices d'Exploration

1. Supposons qu'un utilisateur ait entré les commandes suivantes :

```
$ mkdir "this is a test"
$ ls
this is a test
```

Quelle commande `cd` vous permettrait d'entrer dans ce répertoire ?

```
$ cd this\ is\ a\ test
```

2. Essayez à nouveau, mais après avoir tapé `cd this`, appuyez sur la touche TAB. Qu'est-ce qui est maintenant affiché à l'invite ?

```
$ cd this\ is\ a\ test
```

C'est un exemple d'*autocomplétion*, qui est un outil inestimable non seulement pour gagner du temps, mais aussi pour prévenir les erreurs d'orthographe.

3. Essayez de créer un répertoire dont le nom contient un caractère `\`. Affichez le nom du répertoire avec `ls` et supprimez le répertoire.

Vous pouvez soit échapper la barre oblique inversée en utilisant une autre barre oblique inversée (`\\`), soit utiliser des guillemets simples ou doubles autour de tout le nom du répertoire :

```
$ mkdir my\\dir
$ ls
'my\dir'
$ rmdir 'my\dir'
```



## 2.3 Leçon 2

<b>Certification :</b>	Linux Essentials
<b>Version:</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objectif :</b>	2.3 Utilisation des Répertoires et Listage des Fichiers
<b>Leçon:</b>	2 sur 2

### Introduction

Le système d'exploitation Unix a été conçu à l'origine pour les ordinateurs centraux (mainframe) au milieu des années 60. Ces ordinateurs étaient partagés entre de nombreux utilisateurs, qui accédaient aux ressources du système par l'intermédiaire de *terminaux*. Ces idées fondamentales se retrouvent aujourd'hui dans les systèmes Linux. Nous parlons toujours de l'utilisation de "terminaux" pour entrer des commandes dans le shell, et chaque système Linux est organisé de telle manière qu'il est facile de créer de nombreux utilisateurs sur un seul système.

### Répertoires Personnels

Ceci est un exemple de système de fichiers normal sous Linux :

```
$ tree -L 1 /
/
├─ bin
├─ boot
└─ cdrom
```

```
├─ dev
├─ etc
├─ home
├─ lib
├─ mnt
├─ opt
├─ proc
├─ root
├─ run
├─ sbin
├─ srv
├─ sys
├─ tmp
├─ usr
└─ var
```

La plupart de ces répertoires sont cohérents sur l'ensemble des systèmes Linux. Des serveurs aux supercalculateurs en passant par les minuscules systèmes embarqués, un utilisateur expérimenté de Linux peut être sûr de trouver la commande `ls` dans `/bin`, de modifier la configuration du système en modifiant les fichiers dans `/etc`, et de lire les journaux système dans `/var`. L'emplacement standard de ces fichiers et répertoires est défini par la norme FHS (Filesystem Hierarchy Standard), qui sera abordée dans une leçon ultérieure. Vous en apprendrez plus sur le contenu de ces répertoires au fur et à mesure de votre apprentissage de Linux, mais pour l'instant, sachez que :

- les modifications que vous effectuez dans le système de fichiers racine affecteront tous les utilisateurs, et
- la modification de fichiers dans le système de fichiers racine nécessitera des autorisations d'administrateur.

Cela signifie qu'il sera interdit aux utilisateurs normaux de modifier ces fichiers, et peut-être même de les lire. Nous aborderons le sujet des autorisations dans une section ultérieure.

Nous allons maintenant nous concentrer sur le répertoire `/home`, qui devrait être quelque peu familier à ce stade :

```
$ tree -L 1 /home
/home
├─ user
├─ michael
└─ lara
```

Notre exemple de système comporte trois utilisateurs normaux, et chacun de nos utilisateurs a son propre emplacement dédié, où il peut créer et modifier des fichiers et des répertoires sans affecter son voisin. Par exemple, dans la leçon précédente, nous avons travaillé avec la structure de fichiers suivante :

```
$ tree /home/user
user
├── Documents
│   ├── Mission-Statement
│   └── Reports
│       └── report2018.txt
```

En réalité, le véritable système de fichiers peut ressembler à ceci :

```
$ tree /home
/home
├── user
│   ├── Documents
│   │   ├── Mission-Statement
│   │   └── Reports
│   │       └── report2018.txt
├── michael
│   ├── Documents
│   │   └── presentation-for-clients.odp
│   └── Music
```

...et ainsi de suite pour lara.

Sous Linux, `/home` est similaire à un immeuble d'habitation. De nombreux utilisateurs peuvent y avoir leur espace, séparé en appartements dédiés. Les services publics et la maintenance du bâtiment lui-même sont sous la responsabilité du superintendant utilisateur `root`.

## Le Chemin Relatif Spécial pour le Répertoire Personnel

Lorsque vous démarrez une nouvelle session de terminal sous Linux, vous voyez une invite de commande similaire à celle-ci :

```
user@hostname ~ $
```

Le tilde (`~`) représente ici notre *répertoire personnel*. Si vous exécutez la commande `ls`, vous

verrez une sortie familière :

```
$ cd ~  
$ ls  
Documents
```

Comparez cela avec le système de fichiers ci-dessus pour vérifier votre compréhension.

Examinons maintenant ce que nous savons de Linux : il est semblable à un immeuble d'habitation, avec de nombreux utilisateurs résidant dans `/home`. Le répertoire personnel de l'utilisateur `user` sera donc différent de celui de l'utilisateur `michael`. Pour le démontrer, nous utiliserons la commande `su` (*switch user*) pour changer d'utilisateur.

```
user@hostname ~ $ pwd  
/home/user  
user@hostname ~ $ su - michael  
Password:  
michael@hostname ~ $ pwd  
/home/michael
```

La signification de `~` change en fonction de l'utilisateur. Pour `michael`, le chemin absolu de `~` est `/home/michael`. Pour `lara`, le chemin absolu de `~` est `/home/lara`, et ainsi de suite.

## Chemins des Fichiers Relatifs au Répertoire Personnel

L'utilisation de `~` pour les commandes est très pratique, à condition de ne pas changer d'utilisateur. Nous allons considérer l'exemple suivant pour l'utilisateur `user`, qui a commencé une nouvelle session :

```
$ ls  
Documents  
$ cd Documents  
$ ls  
Mission-Statement  
Reports  
$ cd Reports  
$ ls  
report2018.txt  
$ cd ~  
$ ls
```

## Documents

Notez que les utilisateurs commencent toujours une nouvelle session dans leur répertoire personnel. Dans cet exemple, `user` s'est rendu dans son sous-répertoire `Documents/Reports`, et avec la commande `cd ~`, il est retourné à son point de départ. Vous pouvez effectuer la même action en utilisant la commande `cd` sans argument :

```
$ cd Documents/Reports
$ pwd
/home/user/Documents/Reports
$ cd
$ pwd
/home/user
```

Une dernière chose à noter : nous pouvons spécifier les répertoires personnels des *autres utilisateurs* en précisant le nom d'utilisateur après le tilde. Par exemple :

```
$ ls ~michael
Documents
Music
```

Notez que cela ne fonctionnera que si `michael` nous a donné la permission de consulter le contenu de son répertoire personnel.

Imaginons une situation où `michael` voudrait voir le fichier `report2018.txt` dans le répertoire personnel de `user`. En supposant que `michael` ait la permission de le faire, il peut utiliser la commande `less`.

```
$ less ~user/Documents/Reports/report2018.txt
```

Tout chemin de fichier qui contient le caractère `~` est appelé chemin *relatif au répertoire personnel*.

## Fichiers et Répertoires Cachés

Dans la leçon précédente, nous avons introduit l'option `-a` pour la commande `ls`. Nous avons utilisé `ls -a` pour introduire les deux chemins relatifs spéciaux : `.` et `..`. L'option `-a` permet de lister *tous* les fichiers et répertoires, y compris les fichiers et répertoires *cachés*.



```
$ ls -a ~  
.  
..  
.bash_history  
.bash_logout  
.bash-profile  
.bashrc  
Documents
```

Les fichiers et répertoires cachés commenceront toujours par un point (.). Par défaut, le répertoire personnel d'un utilisateur comprendra de nombreux fichiers cachés. Ceux-ci sont souvent utilisés pour définir des paramètres de configuration spécifiques à l'utilisateur, et ne doivent être modifiés que par un utilisateur expérimenté.

## L'Option de Listage Long

La commande `ls` dispose de nombreuses options pour modifier son comportement. Examinons l'une des options les plus courantes :

```
$ ls -l  
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

`-l` crée un *listage long*. Les fichiers et les répertoires occuperont chacun une ligne, mais des informations supplémentaires sur chaque fichier et répertoire seront affichées.

**-rw-r--r--**

Type de fichier et autorisations du fichier. Notez qu'un fichier normal commencera par un tiret, et qu'un répertoire commencera par `d`.

**1**

Nombre de liens vers le fichier.

**user staff**

Précise la propriété du fichier. `user` est le propriétaire du fichier, et le fichier est également associé au groupe `staff`.

**3606**

Taille du fichier en octets.

Jan 13 2017

Horodatage de la dernière modification du fichier.

report2018.txt

Nom du fichier.

Des sujets tels que la propriété, les autorisations et les liens seront traités dans les prochaines sections. Comme vous pouvez le voir, la version listage long de `ls` est souvent préférable à la version par défaut.

## Options ls Supplémentaires

Vous trouverez ci-dessous quelques-unes des façons les plus courantes d'utiliser la commande `ls`. Comme vous pouvez le voir, l'utilisateur peut combiner de nombreuses options pour obtenir le résultat souhaité.

**ls -lh**

La combinaison d'un *listage long* avec des tailles de fichiers lisibles par l'homme (*human readable*) nous donnera des suffixes utiles tels que `M` pour mégaoctets ou `K` pour kilooctets.

**ls -d \*/**

L'option `-d` permet de lister les répertoires mais pas leur contenu. La combinaison avec `*/` n'affichera que les sous-répertoires et aucun fichier.

**ls -lt**

Combine le *listage long* avec la possibilité de trier par *heure de modification*. Les fichiers comportant les modifications les plus récentes seront en haut, et les fichiers comportant les modifications les plus anciennes seront en bas. Mais cet ordre peut être inversé avec :

**ls -lrt**

Combine le *listage long* avec le *tri par heure (de modification)*, combiné avec `-r` qui *inverse* le tri. Les fichiers avec les modifications les plus récentes se trouvent maintenant en bas de la liste. En plus du tri par *heure de modification*, les fichiers peuvent également être triés par heure d'*accès* ou par heure de *statut*.

**ls -lX**

Combine le *listage long* avec l'option de tri par *eXtension de fichier*. Cela permettra, par exemple, de regrouper tous les fichiers se terminant par `.txt`, tous les fichiers se terminant par `.jpg`, etc.

## ls -S

Le `-S` trie par *taille* de fichier, un peu comme `-t` et `-X` trient respectivement par heure et par extension. Les fichiers les plus volumineux seront classés en premier, et les plus petits en dernier. Notez que le contenu des sous-répertoires *n'est pas* inclus dans le tri.

## ls -R

L'option `-R` modifiera la commande `ls` pour afficher une liste *réursive*. Qu'est-ce que cela signifie ?

## Récursion en Bash

La récursion désigne une situation dans laquelle “quelque chose est défini par rapport à lui-même”. La récursion est un concept très important en informatique, mais sa signification est ici beaucoup plus simple. Examinons notre exemple précédent :

```
$ ls ~  
Documents
```

Nous savons depuis longtemps que `user` a un répertoire personnel, et dans ce répertoire il y a un sous-répertoire. Jusqu'à présent, la commande `ls` ne nous a montré que les fichiers et les sous-répertoires d'un emplacement, mais elle ne peut pas nous dire le contenu de ces sous-répertoires. Dans ces leçons, nous avons utilisé la commande `tree` lorsque nous voulions afficher le contenu de différents répertoires. Malheureusement, `tree` n'est pas l'un des utilitaires de base de Linux et n'est donc pas toujours disponible. Comparez la sortie de `tree` avec la sortie de `ls -R` dans les exemples suivants :

```
$ tree /home/user  
user  
├── Documents  
│   ├── Mission-Statement  
│   └── Reports  
│       └── report2018.txt  
  
$ ls -R ~  
/home/user/:  
Documents  
  
/home/user/Documents:  
Mission-Statement  
Reports
```

```
/home/user/Documents/Reports:  
report2018.txt
```

Comme vous pouvez le voir, avec l'option récursive, nous obtenons une liste de fichiers beaucoup plus longue. En fait, c'est comme si nous lançons la commande `ls` dans le répertoire personnel de `user` et que nous rencontrons un sous-répertoire. Puis, nous sommes entrés dans ce sous-répertoire et avons exécuté la commande `ls` à nouveau. Nous y avons rencontré le fichier `Mission-Statement` et un autre sous-répertoire appelé `Reports`. Nous sommes alors entrés dans ce sous-répertoire et avons exécuté la commande `ls` à nouveau. Essentiellement, exécuter la commande `ls -R` revient à dire à Bash : "Exécute `ls` ici, et répète la commande dans chaque sous-répertoire que tu trouves."

La récursivité est particulièrement importante dans les commandes de modification de fichiers telles que la copie ou la suppression de répertoires. Par exemple, si vous souhaitez copier le sous-répertoire `Documents`, vous devez spécifier une copie récursive afin d'étendre cette commande à tous les sous-répertoires.

## Exercices Guidés

1. Utilisez la structure de fichiers suivante pour répondre aux trois questions suivantes :

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

- Quelle commande permet de naviguer dans le répertoire `network`, quel que soit votre emplacement actuel ?
- Quelle commande `user` peut-il entrer pour naviguer dans son répertoire `Documents` à partir de `/etc/udev` ? Utilisez le chemin le plus court possible.
- Quelle commande `user` peut-il entrer pour naviguer dans le répertoire `Music` de `michael` ? Utilisez le chemin le plus court possible.

2. Considérez la sortie suivante de `ls -lh` pour répondre aux deux questions suivantes. Notez que les répertoires sont indiqués par un `d` en début de ligne.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
```

```
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0067.jpg
-rwxrwxrwx 1 eric eric 1.6M Jul 18 2011 img_0074.jpg
-rwxrwxrwx 1 eric eric 1.8M Jul 18 2011 img_0075.jpg
-rwxrwxrwx 1 eric eric 46K Jul 18 2011 scary.jpg
-rwxrwxrwx 1 eric eric 469K Jan 29 2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx 1 eric eric 498K Jan 29 2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx 1 eric eric 211K Jan 29 2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx 1 eric eric 150K Jul 18 2011 tobermory.jpg
drwxrwxrwx 6 eric eric 4.0K Apr 26 2011 Tokyo/
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 081.jpg
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 085.jpg
-rwxrwxrwx 1 eric eric 944K Jul 18 2011 Toronto 152.jpg
-rwxrwxrwx 1 eric eric 728K Jul 18 2011 Toronto 173.jpg
drwxrwxrwx 2 eric eric 4.0K Jun 5 2016 Wallpapers/
```

- Lorsque vous exécutez la commande `ls -lrS`, quel sera le fichier au début ?

- Veuillez décrire ce que vous attendez comme résultat pour la commande `ls -ad */`.

## Exercices d'Exploration

1. Exécutez la commande `ls -lh` dans un répertoire qui contient des sous-répertoires. Notez la taille indiquée de ces répertoires. Ces tailles de fichiers vous semblent-elles correctes ? Représentent-elles avec précision le contenu de tous les fichiers à l'intérieur de ce répertoire ?
2. Voici une nouvelle commande à essayer : `du -h`. Exécutez cette commande et décrivez la sortie qu'elle vous donne.
3. Sur de nombreux systèmes Linux, vous pouvez taper `ll` et obtenir la même sortie que si vous aviez tapé `ls -l`. Veuillez noter cependant que `ll` n'est *pas* une commande. Par exemple, man `ll` vous donnera le message qu'il n'existe pas d'entrée au manuel pour elle. Ceci est un exemple d'*alias*. Pourquoi les alias pourraient-ils être utiles à un utilisateur ?

# Résumé

Dans cet atelier, vous avez appris :

- que chaque utilisateur de Linux aura un répertoire personnel,
- le répertoire personnel de l'utilisateur courant peut être atteint en utilisant `~`,
- tout chemin de fichier qui utilise `~` est appelé chemin *relatif au répertoire personnel*.

Vous avez également appris certaines des façons les plus courantes de modifier la commande `ls`.

## **-a (all)**

affiche tous les fichiers/répertoires, y compris les fichiers cachés

## **-d (directories)**

liste des répertoires, et non leur contenu

## **-h (human readable)**

affiche les tailles de fichiers dans un format lisible par l'homme

## **-l (long list)**

fournit des détails supplémentaires, un fichier/répertoire par ligne

## **-r (reverse)**

inverse l'ordre d'un tri

## **-R (recursive)**

affiche chaque fichier, y compris les fichiers de chaque sous-répertoire

## **-S (size)**

trie par taille de fichier

## **-t (time)**

trie par heure de modification

## **-X (eXtension)**

trie par extension de fichier



# Réponses aux Exercices Guidés

1. Utilisez la structure de fichiers suivante pour répondre aux trois questions suivantes :

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

- Quelle commande permet de naviguer dans le répertoire `network`, quel que soit votre emplacement actuel ?

```
cd /etc/network
```

- Quelle commande `user` peut-il entrer pour naviguer dans son répertoire `Documents` à partir de `/etc/udev` ? Utilisez le chemin le plus court possible.

```
cd ~/Documents
```

- Quelle commande `user` peut-il entrer pour naviguer dans le répertoire `Music` de `michael` ? Utilisez le chemin le plus court possible :

```
cd ~michael/Music
```

2. Considérez la sortie suivante de `ls -lh` pour répondre aux deux questions suivantes. Notez que les répertoires sont indiqués par un `d` en début de ligne.

```
drwxrwxrwx 5 eric eric 4.0K Apr 26 2011 China/
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0066.jpg
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0067.jpg
-rwxrwxrwx 1 eric eric 1.6M Jul 18 2011 img_0074.jpg
-rwxrwxrwx 1 eric eric 1.8M Jul 18 2011 img_0075.jpg
-rwxrwxrwx 1 eric eric 46K Jul 18 2011 scary.jpg
-rwxrwxrwx 1 eric eric 469K Jan 29 2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx 1 eric eric 498K Jan 29 2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx 1 eric eric 211K Jan 29 2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx 1 eric eric 150K Jul 18 2011 tobermory.jpg
drwxrwxrwx 6 eric eric 4.0K Apr 26 2011 Tokyo/
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 081.jpg
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 085.jpg
-rwxrwxrwx 1 eric eric 944K Jul 18 2011 Toronto 152.jpg
-rwxrwxrwx 1 eric eric 728K Jul 18 2011 Toronto 173.jpg
drwxrwxrwx 2 eric eric 4.0K Jun 5 2016 Wallpapers/
```

- Lorsque vous exécutez la commande `ls -lrS`, quel sera le fichier au début ?

Les trois dossiers sont tous de 4.0K, ce qui est la plus petite taille de fichier. `ls` va ensuite trier les répertoires par ordre alphabétique par défaut. La bonne réponse est le fichier `scary.jpg`.

- Veuillez décrire ce que vous attendez comme résultat pour la commande `ls -ad */`.

Cette commande affichera tous les sous-répertoires, y compris les sous-répertoires cachés.

## Réponses aux Exercices d'Exploration

1. Exécutez la commande `ls -lh` dans un répertoire qui contient des sous-répertoires. Notez la taille indiquée de ces répertoires. Ces tailles de fichiers vous semblent-elles correctes ? Représentent-elles avec précision le contenu de tous les fichiers à l'intérieur de ce répertoire ?

Non, elles ne le font pas. Chaque répertoire a une taille de fichier listé de 4096 octets. C'est parce que les répertoires sont ici une abstraction : ils n'existent pas sous forme d'arborescence sur le disque. Lorsque vous voyez un répertoire listé, vous voyez un *lien* vers une liste de fichiers. La taille de ces liens est de 4096 octets.

2. Voici une nouvelle commande à essayer : `du -h`. Exécutez cette commande et décrivez la sortie qu'elle vous donne.

La commande `du` génère une liste de tous les fichiers et répertoires, et indique la taille de chacun. Par exemple, `du -s` affichera la taille de tous les fichiers, répertoires et sous-répertoires pour un certain emplacement.

3. Sur de nombreux systèmes Linux, vous pouvez taper `ll` et obtenir la même sortie que si vous aviez tapé `ls -l`. Veuillez noter cependant que `ll` n'est *pas* une commande. Par exemple, man `ll` vous donnera le message qu'il n'existe pas d'entrée de manuel pour elle. Qu'est-ce que cela vous suggère à propos d'une caractéristique de la ligne de commande ?

`ll` est un alias de `ls -l`. En Bash, nous pouvons utiliser des alias pour simplifier les commandes les plus courantes. `ll` est souvent défini pour vous dans Linux, mais vous pouvez aussi créer les vôtres.



Linux  
Professional  
Institute

## 2.4 Création, déplacement et suppression de fichiers

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 2.4

### Valeur

2

### Domaines de connaissance les plus importants

- Fichiers et répertoires
- Sensibilité à la casse
- Englobements simples

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- mv, cp, rm, touch
- mkdir, rmdir



## 2.4 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	2 Trouver son Chemin sur un Système Linux
<b>Objectif :</b>	2.4 Création, Déplacement et Suppression de Fichiers
<b>Leçon:</b>	1 sur 1

### Introduction

Cette leçon couvre la gestion des fichiers et des répertoires sous Linux à l'aide d'outils en ligne de commande.

Un fichier est une collection de données avec un nom et un ensemble d'attributs. Si, par exemple, vous deviez transférer certaines photos de votre téléphone à un ordinateur et leur donner des noms descriptifs, vous auriez alors un tas de fichiers d'images sur votre ordinateur. Ces fichiers ont des attributs tels que l'heure du dernier accès au fichier ou de sa dernière modification.

Un répertoire est un type de fichier spécial utilisé pour organiser les fichiers. Une bonne façon de concevoir les répertoires est de les comparer aux dossiers utilisés pour organiser les documents dans une armoire à dossiers. Contrairement aux dossiers papier, vous pouvez facilement placer des répertoires à l'intérieur d'autres répertoires.

La ligne de commande est le moyen le plus efficace de gérer des fichiers sur un système Linux. Le shell et les outils en ligne de commande ont des caractéristiques qui rendent l'utilisation de la ligne de commande plus rapide et plus facile qu'un gestionnaire de fichiers graphique.

Dans cette section, vous utiliserez les commandes `ls`, `mv`, `cp`, `pwd`, `find`, `touch`, `rm`, `rmdir`, `echo`, `cat` et `mkdir` pour gérer et organiser les fichiers et les répertoires.

## Sensibilité à la Casse

Contrairement à Microsoft Windows, les noms de fichiers et de répertoires sur les systèmes Linux sont sensibles à la casse. Cela signifie que les noms `/etc/` et `/ETC/` sont des répertoires différents. Essayez les commandes suivantes :

```
$ cd /
$ ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
$ cd ETC
bash: cd: ETC: No such file or directory
$ pwd
/
$ cd etc
$ pwd
/etc
```

La commande `pwd` vous montre le répertoire dans lequel vous êtes actuellement. Comme vous pouvez le voir, la navigation vers `/ETC` n'a pas fonctionné car il n'existe pas de tel répertoire. La navigation vers le répertoire `/etc` qui existe, a réussi.

## Création de Répertoires

La commande `mkdir` est utilisée pour créer des répertoires.

Créons un nouveau répertoire au sein de notre répertoire personnel :

```
$ cd ~
$ pwd
/home/user
$ ls
Desktop  Documents  Downloads
$ mkdir linux_essentials-2.4
$ ls
Desktop  Documents  Downloads  linux_essentials-2.4
$ cd linux_essentials-2.4
$ pwd
```

```
/home/emma/linux_essentials-2.4
```

Durant cette leçon, toutes les commandes seront exécutées dans ce répertoire ou dans l'un de ses sous-répertoires.

Pour revenir facilement au répertoire de la leçon depuis n'importe quel autre endroit de votre système de fichiers, vous pouvez utiliser la commande :

```
$ cd ~/linux_essentials-2.4
```

Le shell interprète le caractère `~` comme votre répertoire personnel.

Lorsque vous êtes dans le répertoire de la leçon, créez d'autres répertoires que nous utiliserons pour les exercices. Vous pouvez ajouter à `mkdir` tous les noms de répertoires, séparés par des espaces :

```
$ mkdir creating moving copying/files copying/directories deleting/directories
deleting/files globs
mkdir: cannot create directory 'copying/files': No such file or directory
mkdir: cannot create directory 'copying/directories': No such file or directory
mkdir: cannot create directory 'deleting/directories': No such file or directory
mkdir: cannot create directory 'deleting/files': No such file or directory
$ ls
creating globs moving
```

Remarquez le message d'erreur et que seuls les répertoires `moving`, `globs` et `creating` ont été créés. Les répertoires `copying` et `deleting` n'existent pas encore. `mkdir`, par défaut, ne crée pas de répertoire à l'intérieur d'un répertoire qui n'existe pas. L'option `-p` ou `--parents` indique à `mkdir` de créer des répertoires parents s'ils n'existent pas. Essayez la même commande `mkdir` avec l'option `-p` :

```
$ mkdir -p creating moving copying/files copying/directories deleting/directories
deleting/files globs
```

Maintenant, vous n'avez plus de messages d'erreur. Voyons voir quels répertoires existent maintenant :

```
$ find
```

```
.
```

```
./creating
./moving
./globs
./copying
./copying/files
./copying/directories
./deleting
./deleting/directories
./deleting/files
```

Le programme `find` est généralement utilisé pour rechercher des fichiers et des répertoires, mais sans aucune option, il vous montrera la liste de tous les fichiers, répertoires et sous-répertoires de votre répertoire courant.

Les options `-t` et `-r` sont particulièrement pratiques pour lister le contenu d'un répertoire avec `ls`. Elles permettent de trier la sortie par temps (`-t`) et d'inverser l'ordre de tri (`-r`). Dans ce cas, les fichiers les plus récents se trouveront en bas de la sortie.

## Création de Fichiers

En règle générale, les fichiers seront créés par les programmes qui travaillent avec les données stockées dans les fichiers. Un fichier vide peut être créé à l'aide de la commande `touch`. Si vous exécutez la commande `touch` sur un fichier existant, le contenu du fichier ne sera pas modifié, mais l'horodatage de modification des fichiers sera mis à jour.

Exécutez la commande suivante pour créer quelques fichiers pour la leçon de globbing :

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Vérifions maintenant que tous les fichiers existent dans le répertoire `globs` :

```
$ cd globs
$ ls
question1    question14    question23    star1100    star2013
question13   question2012  star10        star2002
```

Vous remarquez comment `touch` a créé les fichiers ? Vous pouvez visualiser le contenu d'un



fichier texte avec la commande `cat`. Essayez-la sur un des fichiers que vous venez de créer :

```
$ cat question14
```

Comme `touch` crée des fichiers vides, vous ne devriez obtenir aucun résultat. Vous pouvez utiliser la commande `echo` avec `>` pour créer des fichiers texte simples. Essayez-la :

```
$ echo hello > question15
$ cat question15
hello
```

`echo` affiche du texte sur la ligne de commande. Le caractère `>` indique au shell d'écrire la sortie d'une commande dans le fichier spécifié au lieu de votre terminal. Cela conduit à ce que la sortie de `echo`, `hello` dans ce cas, soit écrite dans le fichier `question15`. Ceci n'est pas spécifique à `echo`, il peut être utilisé avec n'importe quelle commande.

Soyez prudent lorsque vous utilisez `>` ! Si le fichier cible existe déjà, il sera écrasé !

## Renommage de Fichiers

Les fichiers sont déplacés et renommés avec la commande `mv`.

Définissez `moving` comme votre répertoire de travail :

```
$ cd ~/linux_essentials-2.4/moving
```

Créez des fichiers pour vous entraîner. Vous devriez déjà être familiarisé avec ces commandes :

```
$ touch file1 file22
$ echo file3 > file3
$ echo file4 > file4
$ ls
file1  file22  file3  file4
```

Supposons que `file22` est une faute de frappe et qu'il devrait être le `file2`. Corrigez-la avec la commande `mv`. Lorsque vous renommez un fichier, le premier argument est le nom actuel, le second est le nouveau nom :

```
$ mv file22 file2
$ ls
file1  file2  file3  file4
```

Faites attention à la commande `mv`. Si vous renommez un fichier en lui donnant le nom d'un fichier existant, il l'écrasera. Testons cela avec `file3` et `file4` :

```
$ cat file3
file3
$ cat file4
file4
$ mv file4 file3
$ cat file3
file4
$ ls
file1  file2  file3
```

Remarquez que le contenu de `file3` est maintenant celui de `file4`. Utilisez l'option `-i` pour que `mv` vous alerte si vous êtes sur le point d'écraser un fichier existant. Essayez-la :

```
$ touch file4 file5
$ mv -i file4 file3
mv: overwrite 'file3'? y
```

## Déplacement des Fichiers

Les fichiers sont déplacés d'un répertoire à un autre avec la commande `mv`.

Créez quelques répertoires pour y déplacer des fichiers :

```
$ cd ~/linux_essentials-2.4/moving
$ mkdir dir1 dir2
$ ls
dir1  dir2  file1  file2  file3  file5
```

Déplacez `file1` dans `dir1` :

```
$ mv file1 dir1
$ ls
```

```
dir1 dir2 file2 file3 file5
$ ls dir1
file1
```

Remarquez que le dernier argument de `mv` est le répertoire de destination. Lorsque le dernier argument de `mv` est un répertoire, les fichiers sont déplacés dans celui-ci. Plusieurs fichiers peuvent être spécifiés dans une seule commande `mv` :

```
$ mv file2 file3 dir2
$ ls
dir1 dir2 file5
$ ls dir2
file2 file3
```

Il est également possible d'utiliser `mv` pour déplacer et renommer des répertoires. Renommez `dir1` en `dir3` :

```
$ ls
dir1 dir2 file5
$ ls dir1
file1
$ mv dir1 dir3
$ ls
dir2 dir3 file5
$ ls dir3
file1
```

## Suppression de Fichiers et de Répertoires

La commande `rm` peut supprimer des fichiers et des répertoires, tandis que la commande `rmdir` ne peut supprimer que des répertoires. Nettoyons le répertoire `moving` en supprimant `file5` :

```
$ cd ~/linux_essentials-2.4/moving
$ ls
dir2 dir3 file5
$ rmdir file5
rmdir: failed to remove 'file5': Not a directory
$ rm file5
$ ls
dir2 dir3
```

Par défaut, `rmdir` ne peut supprimer que des répertoires vides, c'est pourquoi nous avons dû utiliser `rm` pour supprimer un fichier régulier. Essayez de supprimer le répertoire `deleting` :

```
$ cd ~/linux_essentials-2.4/
$ ls
copying  creating  deleting  globs  moving
$ rmdir deleting
rmdir: failed to remove 'deleting': Directory not empty
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 files
```

Par défaut, `rmdir` refuse de supprimer un répertoire qui n'est pas vide. Utilisez `rmdir` pour supprimer l'un des sous-répertoires vides du répertoire `deleting` :

```
$ ls -a deleting/files
.  ..
$ rmdir deleting/files
$ ls -l deleting
directories
```

La suppression d'un grand nombre de fichiers ou de profondes structures de répertoires comportant de nombreux sous-répertoires peut sembler fastidieuse, mais elle est en fait facile. Par défaut, `rm` ne fonctionne que sur les fichiers ordinaires. L'option `-r` est utilisée pour passer outre ce comportement. Attention, `rm -r` est un excellent pistolet à pied ! Lorsque vous utilisez l'option `-r`, `rm` ne supprime pas seulement les répertoires, mais tout ce qui se trouve dans ce répertoire, y compris les sous-répertoires et leur contenu. Voyez par vous-même comment `rm -r` fonctionne :

```
$ ls
copying  creating  deleting  globs  moving
$ rm deleting
rm: cannot remove 'deleting': Is a directory
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
$ rm -r deleting
$ ls
copying  creating  globs  moving
```

Vous avez remarqué que `deleting` a disparu, même si elle n'était pas vide ? Comme `mv`, `rm` dispose d'une option `-i` pour vous alerter avant toute action. Utilisez `rm -ri` pour supprimer les répertoires de la section `moving` qui ne sont plus nécessaires :

```
$ find
.
./creating
./moving
./moving/dir2
./moving/dir2/file2
./moving/dir2/file3
./moving/dir3
./moving/dir3/file1
./globs
./globs/question1
./globs/question2012
./globs/question23
./globs/question13
./globs/question14
./globs/star10
./globs/star1100
./globs/star2002
./globs/star2013
./globs/question15
./copying
./copying/files
./copying/directories
$ rm -ri moving
rm: descend into directory 'moving'? y
rm: descend into directory 'moving/dir2'? y
rm: remove regular empty file 'moving/dir2/file2'? y
rm: remove regular empty file 'moving/dir2/file3'? y
rm: remove directory 'moving/dir2'? y
rm: descend into directory 'moving/dir3'? y
rm: remove regular empty file 'moving/dir3/file1'? y
rm: remove directory 'moving/dir3'? y
rm: remove directory 'moving'? y
```

## Copie de Fichiers et de Répertoires

La commande `cp` est utilisée pour copier des fichiers et des répertoires. Copiez quelques fichiers dans le répertoire `copying` :

```
$ cd ~/linux_essentials-2.4/copying
$ ls
directories  files
$ cp /etc/nsswitch.conf files/nsswitch.conf
$ cp /etc/issue /etc/hostname files
```

Si le dernier argument est un répertoire, `cp` créera une copie des arguments précédents à l'intérieur de ce répertoire. Comme `mv`, plusieurs fichiers peuvent être spécifiés en même temps, à condition que la cible soit un répertoire.

Lorsque les deux opérandes de `cp` sont des fichiers et que les deux fichiers existent, `cp` écrase le second fichier avec une copie du premier fichier. Pratiquons cela en écrasant le fichier `issue` avec le fichier `hostname` :

```
$ cd ~/linux_essentials-2.4/copying/files
$ ls
hostname  issue  nsswitch.conf
$ cat hostname
mycomputer
$ cat issue
Debian GNU/Linux 9 \n \l

$ cp hostname issue
$ cat issue
mycomputer
```

Essayons maintenant de créer une copie du répertoire `files` dans le répertoire `directories` :

```
$ cd ~/linux_essentials-2.4/copying
$ cp files directories
cp: omitting directory 'files'
```

Comme vous pouvez le constater, `cp` ne fonctionne par défaut que sur des fichiers individuels. Pour copier un répertoire, vous utilisez l'option `-r`. Gardez à l'esprit que l'option `-r` fera que `cp` copie également le contenu du répertoire que vous copiez :

```
$ cp -r files directories
$ find
.
./files
```

```
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
```

Remarquez comment, lorsqu'un répertoire existant a été utilisé comme destination, `cp` crée une copie du répertoire source à l'intérieur de celui-ci ? Si la destination n'existe pas, il la crée et la remplit avec le contenu du répertoire source :

```
$ cp -r files files2
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
./files2
./files2/nsswitch.conf
./files2/fstab
./files2/hostname
```

## Globbing

Ce que l'on appelle communément le globbing est un langage simple de correspondance de motifs. Les shells en ligne de commande sur les systèmes Linux utilisent ce langage pour désigner des groupes de fichiers dont les noms correspondent à un motif spécifique. POSIX.1-2017 spécifie les caractères suivants pour la correspondance des motifs :

**\***

Correspond à un nombre quelconque de caractères, y compris aucun caractère

?

Correspond à n'importe quel caractère une fois

[]

Correspond à une classe de caractères

En anglais, cela signifie que vous pouvez dire à votre shell de correspondre à un modèle plutôt qu'à une chaîne de texte littérale. Habituellement, les utilisateurs de Linux spécifient plusieurs fichiers avec un glob au lieu de taper le nom de chaque fichier. Exécutez les commandes suivantes :

```
$ cd ~/linux_essentials-2.4/globs
$ ls
question1    question14  question2012  star10      star2002
question13   question15  question23    star1100    star2013
$ ls star1*
star10  star1100
$ ls star*
star10  star1100  star2002  star2013
$ ls star2*
star2002  star2013
$ ls star2*2
star2002
$ ls star2013*
star2013
```

Le shell étend `*` à n'importe quel nombre de caractères, donc votre shell interprète `star*` comme signifiant tout ce qui, dans le contexte pertinent, commence par `star`. Lorsque vous exécutez la commande `ls star*`, votre shell n'exécute pas le programme `ls` avec l'argument `star*`, il recherche les fichiers dans le répertoire courant qui correspondent au motif `star*` (y compris juste `star`), et transforme chaque fichier correspondant au motif en un argument de `ls` :

```
$ ls star*
```

en ce qui concerne `ls` est l'équivalent de

```
$ ls star10  star1100  star2002  star2013
```

Le caractère `*` ne signifie rien pour `ls`. Pour le prouver, exécutez la commande suivante :



```
$ ls star\*
ls: cannot access star*: No such file or directory
```

Lorsque vous faites précéder un caractère d'un `\`, vous demandez à votre shell de ne pas l'interpréter. Dans ce cas, vous voulez que `ls` ait un argument `star*` à la place de ce que le glob `star*` développe.

Le caractère `?` s'étend à n'importe quel caractère une fois. Essayez les commandes suivantes pour vous en rendre compte par vous-même :

```
$ ls
question1  question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls question?
question1
$ ls question1?
question13  question14  question15
$ ls question?3
question13  question23
$ ls question13?
ls: cannot access question13?: No such file or directory
```

Les crochets `[]` sont utilisés pour faire correspondre des plages ou des classes de caractères. Les crochets `[]` fonctionnent comme dans les expressions régulières POSIX, mais avec les globs, le `^` est utilisé à la place du `!`.

Créez des fichiers pour faire des expériences :

```
$ mkdir brackets
$ cd brackets
$ touch file1 file2 file3 file4 filea fileb filec file5 file6 file7
```

Les plages entre les crochets `[]` sont exprimées par un `-` :

```
$ ls
file1 file2 file3 file4 file5 file6 file7 filea fileb filec
$ ls file[1-2]
file1 file2
$ ls file[1-3]
file1 file2 file3
```

Plusieurs plages peuvent être spécifiées :

```
$ ls file[1-25-7]
file1 file2 file5 file6 file7
$ ls file[1-35-6a-c]
file1 file2 file3 file5 file6 filea fileb filec
```

Les crochets peuvent également être utilisés pour correspondre à un ensemble spécifique de caractères.

```
$ ls file[1a5]
file1 file5 filea
```

Vous pouvez également utiliser le caractère `^` comme premier caractère pour tout faire correspondre sauf certains caractères.

```
$ ls file[^a]
file1 file2 file3 file4 file5 file6 file7 fileb filec
```

La dernière chose que nous aborderons dans cette leçon, ce sont les classes de caractères. Pour correspondre à une classe de caractères, vous utilisez `[ :nomclasse: ]`. Par exemple, pour utiliser la classe de chiffres, qui correspond aux chiffres, vous feriez quelque chose comme ceci :

```
$ ls file[[:digit:]]
file1 file2 file3 file4 file5 file6 file7
$ touch file1a file11
$ ls file[[:digit:]]a
file1 file2 file3 file4 file5 file6 file7 filea
$ ls file[[:digit:]]a
file1a
```

Le glob `file[[:digit:]]a`, correspond à `file` suivi d'un chiffre ou d'un `a`.

Les classes de caractères prises en charge dépendent de votre région actuelle. POSIX requiert les classes de caractères suivantes pour toutes les régions :

**`[ :alnum: ]`**

Lettres et chiffres.

**[ :alpha: ]**

Lettres majuscules ou minuscules.

**[ :blank: ]**

Espaces et tabulations.

**[ :cntrl: ]**

Caractères de contrôle, par exemple : retour arrière, bell, NAK, Echap.

**[ :digit: ]**

Chiffres (0123456789).

**[ :graph: ]**

Les caractères graphiques (tous les caractères sauf `ctrl` et le caractère espace)

**[ :lower: ]**

Lettres minuscules (a-z).

**[ :print: ]**

Les caractères imprimables (alnum, `ponct` et le caractère espace).

**[ :punct: ]**

Les caractères de ponctuation, c'est-à-dire `!`, `&`, `"`.

**[ :space: ]**

Les caractères d'espacement, par exemple les tabulations, les espaces, les lignes de rappel.

**[ :upper: ]**

Lettres majuscules (A-Z).

**[ :xdigit: ]**

Chiffres hexadécimaux (généralement 0123456789abcdefABCDEF).

# Exercices Guidés

1. Compte tenu de ce qui suit, sélectionnez les répertoires qui seraient créés par la commande `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

/tmp	
/tmp/outfiles	
/tmp/outfiles/text	
/tmp/outfiles/text/today	
/tmp/infiles	
/tmp/infiles/text	
/tmp/infiles/text/today	

2. Que fait l'option `-v` pour `mkdir`, `rm`, et `cp` ?

3. Que se passe-t-il si vous tentez accidentellement de copier trois fichiers sur la même ligne de commande vers un fichier qui existe déjà au lieu d'un répertoire ?

4. Que se passe-t-il lorsque vous utilisez `mv` pour déplacer un répertoire dans lui-même ?

5. Comment supprimer tous les fichiers de votre répertoire courant qui commencent par `old` ?

6. Lequel des fichiers suivants correspondrait à `log_[a-z]201?*_01.txt` ?

log_3_2017_Jan_01.txt	
log+_2017_Feb_01.txt	

log_b_2007_Mar_01.txt	
log_f_201A_Wednesday_01.txt	

7. Créez quelques globs pour correspondre à la liste suivante de noms de fichiers :

doc100

doc200

doc301

doc401

## Exercices d'Exploration

1. Utilisez la page de manuel de `cp` pour savoir comment faire une copie d'un fichier et faire correspondre les autorisations et le temps de modification à l'original.

2. Que fait la commande `rmdir -p` ? Expérimentez-la et expliquez en quoi elle diffère de `rm -r`.

3. N'EXÉCUTEZ PAS CETTE COMMANDE : Que pensez-vous que le `rm -ri /*` fera ? (HONNÊTEMENT, N'ESSAYEZ PAS DE LE FAIRE !)

4. Outre l'utilisation de `-i`, est-il possible d'éviter que `mv` n'écrase les fichiers de destination ?

5. Expliquez la commande `cp -u`.

## Résumé

L'environnement de ligne de commande Linux fournit des outils pour gérer les fichiers. Parmi les plus utilisés, citons `cp`, `mv`, `mkdir`, `rm` et `rmdir`. Ces outils, combinés avec les globs, permettent aux utilisateurs de faire beaucoup de travaux très rapidement.

De nombreuses commandes ont une option `-i`, qui vous alerte avant de faire quelque chose. Cette option peut vous épargner bien des soucis si vous avez fait une faute de frappe.

Beaucoup de commandes ont une option `-r`. L'option `-r` signifie généralement récursion. En mathématiques et en informatique, une fonction récursive est une fonction qui s'utilise elle-même dans sa définition. En ce qui concerne les outils en ligne de commande, cela signifie généralement appliquer la commande à un répertoire et à tout ce qui s'y trouve.

Commandes utilisées dans cette leçon :

### **cat**

Lie et affiche le contenu d'un fichier.

### **cp**

Copie des fichiers ou des répertoires.

### **echo**

Affiche une chaîne de caractères.

### **find**

Parcoure une arborescence de système de fichiers et recherche les fichiers correspondant à un ensemble de critères spécifiques.

### **ls**

Affiche les propriétés des fichiers et des répertoires et énumère le contenu d'un répertoire.

### **mkdir**

Crée de nouveaux répertoires.

### **mv**

Déplace ou renomme des fichiers ou des répertoires.

### **pwd**

Affiche le répertoire de travail actuel.

**rm**

Supprime des fichiers ou des répertoires.

**rmdir**

Supprime des répertoires.

**touch**

Crée de nouveaux fichiers vides ou met à jour l'horodatage de modification d'un fichier existant.



## Réponses aux Exercices Guidés

1. Compte tenu de ce qui suit, sélectionnez les répertoires qui seraient créés par la commande `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

Les répertoires marqués seraient créés. Les répertoires `/tmp`, `/tmp/outfiles`, et `/tmp/outfiles/text` existent déjà, donc `mkdir` les ignorera.

<code>/tmp</code>	
<code>/tmp/outfiles</code>	
<code>/tmp/outfiles/text</code>	
<code>/tmp/outfiles/text/today</code>	X
<code>/tmp/infiles</code>	X
<code>/tmp/infiles/text</code>	X
<code>/tmp/infiles/text/today</code>	X

2. Que fait l'option `-v` pour `mkdir`, `rm`, et `cp` ?

Généralement, l'option `-v` active le fonctionnement verbeux. Elle fait en sorte que les programmes respectifs affichent ce qu'ils font quand ils le font :

```
$ rm -v a b
removed 'a'
removed 'b'
$ mv -v a b
'a' -> 'b'
$ cp -v b c
'b' -> 'c'
```

3. Que se passe-t-il si vous tentez accidentellement de copier trois fichiers sur la même ligne de commande vers un fichier qui existe déjà au lieu d'un répertoire ?

`cp` refusera de faire quoi que ce soit et émettra un message d'erreur :

```
$ touch a b c d
$ cp a b c d
cp: target 'd' is not a directory
```

4. Que se passe-t-il lorsque vous utilisez `mv` pour déplacer un répertoire dans lui-même ?

Vous recevrez un message d'erreur vous indiquant que `mv` ne peut pas faire cela.

```
$ mv a a
mv: cannot move 'a' to a subdirectory of itself, 'a/a'
```

5. Comment supprimer tous les fichiers de votre répertoire courant qui commencent par `old` ?

You would use the glob `old*` with `rm`:

```
$ rm old*
```

6. Lequel des fichiers suivants correspondrait à `log_[a-z]201?*_01.txt` ?

<code>log_3_2017_Jan_01.txt</code>	
<code>log+_2017_Feb_01.txt</code>	
<code>log_b_2007_Mar_01.txt</code>	
<code>log_f_201A_Wednesday_01.txt</code>	X

```
$ ls log_[a-z]_201?*_01.txt
log_f_201A_Wednesday_01.txt
```

`log_[a-z]` correspond à `log_` suivi de n'importe quelle lettre minuscule, donc `log_f_201A_Wednesday_01.txt` et `log_b_2007_Mar_01.txt` correspondent tous deux. `_201?` correspond à n'importe quel caractère une fois, donc seul `log_f_201A_Wednesday_01.txt` correspond. Enfin, `*_01.txt` correspond à tout ce qui se termine par `_01.txt`, donc notre option restante correspond.

7. Créez quelques globs pour correspondre à la liste suivante de noms de fichiers :

```
doc100
```

```
doc200  
doc301  
doc401
```

Il existe plusieurs solutions. En voici quelques-unes :

```
doc*  
doc[1-4]*  
doc?0?  
doc[1-4]0?
```

## Réponses aux Exercices d'Exploration

1. Utilisez la page de manuel de `cp` pour savoir comment faire une copie d'un fichier et faire correspondre les autorisations et le temps de modification à l'original.

Vous utiliseriez l'option `-p`. Portion de la page de manuel :

```
$ man cp
-p      same as --preserve=mode,ownership,timestamps
--preserve[=ATTR_LIST]
        preserve the specified attributes (default: mode,ownership,time-
        stamps), if possible additional attributes: context, links,
        xattr, all
```

2. Que fait la commande `rmdir -p` ? Expérimentez-la et expliquez en quoi elle diffère de `rm -r`.

Elle fait en sorte que `rmdir` se comporte comme `mkdir -p`. Si on lui passe une arborescence de répertoires vides, elle les supprime tous.

```
$ find
.
./a
./a/b
./a/b/c
$ rmdir -p a/b/c
$ ls
```

3. N'EXÉCUTEZ PAS CETTE COMMANDE : Que pensez-vous que `rm -ri /*` fera ? (SÉRIEUSEMENT, N'ESSAYEZ PAS DE LA FAIRE !)

Il supprimera tous les fichiers et répertoires accessibles en écriture par votre compte utilisateur. Cela inclut tous les systèmes de fichiers en réseau.

4. Outre l'utilisation de `-i`, est-il possible d'éviter que `mv` n'écrase les fichiers de destination ?

Oui, l'option `-n` ou `--no-clobber` empêche `mv` d'écraser les fichiers.

```
$ cat a
a
$ cat b
b
```

```
$ mv -n a b
$ cat b
b
```

## 5. Expliquer `cp -u`.

L'option `-u` fait en sorte que `cp` ne copie un fichier que si la destination est manquante ou plus ancienne que le fichier source.

```
$ ls -l
total 24K
drwxr-xr-x 123 emma student 12K Feb  2 05:34 ..
drwxr-xr-x  2 emma student 4.0K Feb  2 06:56 .
-rw-r--r--  1 emma student  2 Feb  2 06:56 a
-rw-r--r--  1 emma student  2 Feb  2 07:00 b
$ cat a
a
$ cat b
b
$ cp -u a b
$ cat b
b
$ cp -u a c
$ ls -l
total 12
-rw-r--r-- 1 emma student 2 Feb  2 06:56 a
-rw-r--r-- 1 emma student 2 Feb  2 07:00 b
-rw-r--r-- 1 emma student 2 Feb  2 07:00 c
```



**Linux  
Professional  
Institute**

## **Sujet 3 : Le pouvoir de la ligne de commande**



## 3.1 Archivage de fichiers en ligne de commande

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 3.1

### Valeur

2

### Domaines de connaissance les plus importants

- Fichiers, répertoires
- Archives, compression

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `tar`
- Options courantes de `tar`
- `gzip`, `bzip2`, `xz`
- `zip`, `unzip`



## 3.1 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	3 La Puissance de la Ligne de Commande
<b>Objectif :</b>	3.1 Archivage des Fichiers depuis la Ligne de Commande
<b>Leçon :</b>	1 sur 1

### Introduction

La compression est utilisée pour réduire la quantité d'espace qu'un ensemble spécifique de données consomme. La compression est couramment utilisée pour réduire l'espace nécessaire au stockage d'un fichier. Une autre utilisation courante consiste à réduire la quantité de données envoyées via une connexion réseau.

La compression fonctionne en remplaçant les modèles répétitifs dans les données. Supposons que vous ayez un roman. Certains mots sont extrêmement courants et comportent plusieurs caractères, comme le mot "le". Vous pourriez réduire considérablement la taille du roman si vous remplaciez ces mots et motifs communs à plusieurs caractères par des remplacements à un seul caractère. Par exemple, remplacez "le" par une lettre grecque qui n'est pas utilisée ailleurs dans le texte. Les algorithmes de compression de données sont similaires à celui-ci mais plus complexes.

Il existe deux types de compression : *sans perte* et *avec perte*. Les objets compressés avec un algorithme sans perte peuvent être décompressés pour retrouver leur forme originale. Les données compressées avec un algorithme avec perte ne peuvent pas être récupérées. Les algorithmes avec perte sont souvent utilisés pour les images, la vidéo et le son lorsque la perte de



qualité est imperceptible pour l'homme, sans rapport avec le contexte, ou lorsque la perte vaut l'espace de stockage ou le débit du réseau économisé.

Les outils d'archivage sont utilisés pour regrouper les fichiers et les répertoires en un seul fichier. Les sauvegardes, le regroupement du code source des logiciels et la conservation des données en sont des utilisations courantes.

L'archivage et la compression sont couramment utilisés ensemble. Certains outils d'archivage compressent même leur contenu par défaut. D'autres peuvent compresser leur contenu de manière optionnelle. Certains outils d'archivage doivent être utilisés conjointement avec des outils de compression autonomes si vous souhaitez compresser le contenu.

L'outil le plus courant pour l'archivage de fichiers sur les systèmes Linux est `tar`. La plupart des distributions Linux sont livrées avec la version GNU de `tar`, c'est donc celui qui sera abordé dans cette leçon. `tar` gère seulement l'archivage des fichiers mais ne les compresse pas.

Il existe de nombreux outils de compression sur Linux. Parmi les outils de compression sans perte les plus courants, on trouve `bzip2`, `gzip` et `xz`. Vous les trouverez tous les trois sur la plupart des systèmes. Vous pouvez rencontrer un système ancien ou très minimal où `xz` ou `bzip` ne sont pas installés. Si vous devenez un utilisateur régulier de Linux, vous rencontrerez probablement des fichiers compressés avec ces trois outils. Tous trois utilisent des algorithmes différents, de sorte qu'un fichier compressé avec un outil ne peut pas être décompressé par un autre. Les outils de compression ont un compromis à faire. Si vous souhaitez un taux de compression élevé, il vous faudra plus de temps pour compresser et décompresser le fichier. En effet, une compression plus élevée nécessite plus de travail pour trouver des modèles plus complexes. Tous ces outils compressent les données mais ne peuvent pas créer d'archives contenant plusieurs fichiers.

Les outils de compression autonomes ne sont généralement pas disponibles sur les systèmes Windows. Les outils d'archivage et de compression de Windows sont généralement regroupés. Gardez cela à l'esprit si vous avez des systèmes Linux et Windows qui ont besoin de partager des fichiers.

Les systèmes Linux disposent également d'outils pour traiter les fichiers `.zip` couramment utilisés sur le système Windows. Ces outils sont appelés `zip` et `unzip`. Ces outils ne sont pas installés par défaut sur tous les systèmes, donc si vous avez besoin de les utiliser, vous devrez peut-être les installer. Heureusement, on les trouve généralement dans les dépôts de paquets des distributions.

## Les Outils de Compression

L'espace disque économisé par la compression des fichiers dépend de plusieurs facteurs. La

nature des données que vous compressez, l'algorithme utilisé pour compresser les données et le niveau de compression. Tous les algorithmes ne prennent pas en charge différents niveaux de compression.

Commençons par mettre en place quelques fichiers de test à compresser :

```
$ mkdir ~/linux_essentials-3.1
$ cd ~/linux_essentials-3.1
$ mkdir compression archiving
$ cd compression
$ cat /etc/* > bigfile 2> /dev/null
```

Nous créons maintenant trois copies de ce fichier :

```
$ cp bigfile bigfile2
$ cp bigfile bigfile3
$ cp bigfile bigfile4
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile4
```

Nous allons maintenant compresser les fichiers avec les différents outils de compression susmentionnés :

```
$ bzip2 bigfile2
$ gzip bigfile3
$ xz bigfile4
$ ls -lh
total 1.2M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 170K Jun 23 08:08 bigfile2.bz2
-rw-r--r-- 1 emma emma 179K Jun 23 08:08 bigfile3.gz
-rw-r--r-- 1 emma emma 144K Jun 23 08:08 bigfile4.xz
```

Comparez la taille des fichiers compressés à celle du fichier non compressé appelé `bigfile`. Remarquez également comment les outils de compression ont ajouté des extensions aux noms de fichiers et supprimé les fichiers non compressés.

Utilisez `bunzip2`, `gunzip`, ou `unxz` pour décompresser les fichiers :

```
$ bunzip2 bigfile2.bz2
$ gunzip bigfile3.gz
$ unxz bigfile4.xz
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile4
```

Remarquez à nouveau que le fichier compressé est maintenant supprimé une fois qu'il est décompressé.

Certains outils de compression prennent en charge différents niveaux de compression. Un niveau de compression plus élevé nécessite généralement plus de mémoire et de cycles CPU, mais se traduit par un fichier compressé plus petit. L'inverse est vrai pour un niveau inférieur. Vous trouverez ci-dessous une démonstration avec `xz` et `gzip` :

```
$ cp bigfile bigfile-gz1
$ cp bigfile bigfile-gz9
$ gzip -1 bigfile-gz1
$ gzip -9 bigfile-gz9
$ cp bigfile bigfile-xz1
$ cp bigfile bigfile-xz9
$ xz -1 bigfile bigfile-xz1
$ xz -9 bigfile bigfile-xz9
$ ls -lh bigfile bigfile-* *
total 3.5M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 205K Jun 23 13:14 bigfile-gz1.gz
-rw-r--r-- 1 emma emma 178K Jun 23 13:14 bigfile-gz9.gz
-rw-r--r-- 1 emma emma 156K Jun 23 08:08 bigfile-xz1.xz
-rw-r--r-- 1 emma emma 143K Jun 23 08:08 bigfile-xz9.xz
```

Il n'est pas nécessaire de décompresser un fichier chaque fois que vous l'utilisez. Les outils de compression sont généralement fournis avec des versions spéciales d'outils courants utilisés pour lire les fichiers texte. Par exemple, `gzip` possède une version de `cat`, `grep`, `diff`, `less`, `more` et quelques autres. Pour `gzip`, les outils sont préfixés par un `z`, alors que `bzip2` les préfixe avec `bz` et `xz` les préfixe avec `xz`. Voici un exemple d'utilisation de `zcat` pour lire et afficher un fichier

compressé avec `gzip` :

```
$ cp /etc/hosts ./
$ gzip hosts
$ zcat hosts.gz
127.0.0.1    localhost

# The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

## Les Outils d'Archivage

Le programme `tar` est probablement l'outil d'archivage le plus utilisé sur les systèmes Linux. Au cas où vous demanderiez pourquoi il est nommé ainsi, il s'agit d'une abréviation de "tape archive" ("archive sur bande"). Les fichiers créés avec `tar` sont souvent appelés *tar balls*. Il est très courant que les applications distribuées sous forme de code source soient dans des tar balls.

La version GNU de `tar` que les distributions Linux livrent a beaucoup d'options. Cette leçon va couvrir le sous-ensemble le plus couramment utilisé.

Commençons par créer une archive des fichiers utilisés pour la compression :

```
$ cd ~/linux_essentials-3.1
$ tar cf archiving/3.1.tar compression
```

L'option `c` indique à `tar` de créer un nouveau fichier d'archive et l'option `f` est le nom du fichier à créer. L'argument qui suit immédiatement les options sera toujours le nom du fichier sur lequel travailler. Les autres arguments sont les chemins d'accès aux fichiers ou répertoires que vous souhaitez ajouter, lister ou extraire du fichier. Dans l'exemple, nous ajoutons le répertoire `compression` et tout son contenu à l'archive.

Pour visualiser le contenu d'une tar ball, utilisez l'option `t` de `tar` :

```
$ tar -tf 3.1.tar
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
```

```
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
```

Remarquez comment les options sont précédées de `-`. Contrairement à la plupart des programmes, avec `tar`, le `-` est facultatif lors de la spécification des options, bien qu'il ne cause aucun dommage s'il est utilisé.

**NOTE**

Vous pouvez utiliser l'option `-v` pour permettre à `tar` d'afficher les noms des fichiers sur lesquels il opère lors de la création ou de l'extraction d'une archive.

Maintenant, extrayons le fichier :

```
$ cd ~/linux_essentials-3.1/archiving
$ ls
3.1.tar
$ tar xf 3.1.tar
$ ls
3.1.tar  compression
```

Supposons que vous n'ayez besoin que d'un seul fichier de l'archive. Si c'est le cas, vous pouvez le spécifier après le nom du fichier de l'archive. Vous pouvez spécifier plusieurs fichiers si nécessaire :

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ ls
3.1.tar
$ tar xvf 3.1.tar compression/hosts.gz
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
```

```
$ ls
3.1.tar  compression
$ ls compression
hosts.gz
```

À l'exception des chemins absolus (chemins commençant par /), les fichiers `tar` conservent l'intégralité du chemin d'accès aux fichiers lors de leur création. Comme le fichier `3.1.tar` a été créé avec un seul répertoire, ce répertoire sera créé par rapport à votre répertoire de travail actuel lors de l'extraction. Un autre exemple devrait permettre de clarifier ce point :

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ cd ../compression
$ tar cf ../tar/3.1-nodir.tar *
$ cd ../archiving
$ mkdir untar
$ cd untar
$ tar -xf ../3.1-nodir.tar
$ ls
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
```

Si vous souhaitez utiliser le chemin absolu dans un fichier `tar`, vous devez utiliser l'option `P`. Sachez que cela peut écraser des fichiers importants et provoquer des erreurs sur votre système.

Le programme `tar` peut également gérer la compression et la décompression des archives à la volée. `tar` le fait en appelant l'un des outils de compression mentionnés plus haut dans cette section. Il suffit d'ajouter l'option appropriée à l'algorithme de compression. Les plus couramment utilisés sont `j`, `J` et `z` pour `bzip2`, `xz` et `gzip`, respectivement. Vous trouverez ci-dessous des exemples utilisant les algorithmes susmentionnés :

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
$ tar -czf gzip.tar.gz bigfile bigfile2 bigfile3
$ tar -cjf bzip2.tar.bz2 bigfile bigfile2 bigfile3
$ tar -cJf xz.tar.xz bigfile bigfile2 bigfile3
$ ls -l | grep tar
```

```
-rw-r--r-- 1 emma emma 450202 Jun 27 05:56 bzip2.tar.bz2
-rw-r--r-- 1 emma emma 548656 Jun 27 05:55 gzip.tar.gz
-rw-r--r-- 1 emma emma 147068 Jun 27 05:56 xz.tar.xz
```

Remarquez que dans l'exemple, les fichiers `.tar` ont des tailles différentes. Cela montre qu'ils ont été compressés avec succès. Si vous créez des archives `.tar` compressées, vous devez toujours ajouter une deuxième extension de fichier indiquant l'algorithme que vous avez utilisé. Il s'agit de `.xz`, `.bz` et `.gz` pour `xz`, `bzip2` et `gzip`, respectivement. Parfois, des extensions raccourcies telles que `.tgz` sont utilisées.

Il est possible d'ajouter des fichiers aux archives de tar non compressées déjà existantes. Pour ce faire, utilisez l'option `u`. Si vous essayez d'ajouter un fichier à une archive compressée, vous obtiendrez une erreur.

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  bzip2.tar.bz2  hosts.gz
bigfile2 bigfile4  bigfile-gz9.gz  bigfile-xz9.xz  gzip.tar.gz    xz.tar.xz
$ tar cf plain.tar bigfile bigfile2 bigfile3
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
$ tar uf plain.tar bigfile4
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
bigfile4
$ tar uzf gzip.tar.gz bigfile4
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
```

## La Gestion des fichiers ZIP

Les machines Windows n'ont souvent pas d'applications pour gérer les tar balls ou de nombreux outils de compression que l'on trouve couramment sur les systèmes Linux. Si vous avez besoin d'interagir avec les systèmes Windows, vous pouvez utiliser des fichiers ZIP. Un fichier ZIP est un fichier d'archive similaire à un fichier `tar` compressé.

Les programmes `zip` et `unzip` peuvent être utilisés pour travailler avec des fichiers ZIP sur les

systèmes Linux. L'exemple ci-dessous devrait vous suffire pour commencer à les utiliser. Nous commençons par créer un ensemble de fichiers :

```
$ cd ~/linux_essentials-3.1
$ mkdir zip
$ cd zip/
$ mkdir dir
$ touch dir/file1 dir/file2
```

Nous utilisons maintenant le format `zip` pour regrouper ces fichiers dans un fichier ZIP :

```
$ zip -r zipfile.zip dir
  adding: dir/ (stored 0%)
  adding: dir/file1 (stored 0%)
  adding: dir/file2 (stored 0%)
$ rm -rf dir
```

Enfin, nous décompressons à nouveau le fichier ZIP :

```
$ ls
zipfile.zip
$ unzip zipfile.zip
Archive:  zipfile.zip
   creating: dir/
  extracting: dir/file1
  extracting: dir/file2
$ find
.
./zipfile.zip
./dir
./dir/file1
./dir/file2
```

Pour l'ajout de répertoires à des fichiers ZIP, l'option `-r` fait en sorte que le `zip` inclue le contenu d'un répertoire. Sans cette option, vous auriez un répertoire vide dans le fichier ZIP.



## Exercices Guidés

1. D'après les extensions, lesquels des outils suivants ont été utilisés pour créer ces fichiers ?

Nom du fichier	tar	gzip	bzip2	xz
archive.tar				
archive.tgz				
archive.tar.xz				

2. D'après les extensions, lesquels de ces fichiers sont des archives et lesquels sont compressés ?

Nom du fichier	Archive	Compressé
file.tar		
file.tar.bz2		
file.zip		
file.xz		

3. Comment ajouter un fichier à une archive tar compressés au format gzip ?

4. Quelle option de tar demande à une archive tar d'inclure les chemins absolus commençant par / ?

5. L'outil zip supporte-t-il différents niveaux de compression ?

## Exercices d'Exploration

1. Lors de l'extraction de fichiers, est-ce que `tar` supporte les globs dans la liste des fichiers ?
2. Comment s'assurer qu'un fichier décompressé est identique au fichier avant qu'il ne soit compressé ?
3. Que se passe-t-il si vous essayez d'extraire un fichier d'une archive `tar` qui existe déjà sur votre système de fichiers ?
4. Comment extraire le fichier `archive.tgz` sans utiliser l'option `tar z` ?

## Résumé

Les systèmes Linux disposent de plusieurs outils de compression et d'archivage. Cette leçon a porté sur les plus courants. L'outil d'archivage le plus courant est `tar`. Si une interaction avec les systèmes Windows est nécessaire, `zip` et `unzip` peuvent créer et extraire des fichiers ZIP.

La commande `tar` a quelques options qui valent la peine d'être mémorisées. Il s'agit de `x` pour extraire, `c` pour créer, `t` pour afficher le contenu et `u` pour ajouter ou remplacer des fichiers. L'option `v` affiche les fichiers qui sont traités par `tar` lors de la création ou de l'extraction d'une archive.

Le dépôt d'une distribution Linux typique comporte de nombreux outils de compression. Les plus courants sont `gzip`, `bzip2` et `xz`. Les algorithmes de compression prennent souvent en charge différents niveaux qui vous permettent d'optimiser la vitesse ou la taille des fichiers. Les fichiers peuvent être décompressés avec `gunzip`, `bunzip2` et `unxz`.

Les outils de compression ont généralement des programmes qui se comportent comme les outils de fichiers texte courants, à la différence qu'ils fonctionnent sur des fichiers compressés. Quelques-uns d'entre eux sont `zcat`, `bzcat` et `xzcat`. Les outils de compression sont généralement livrés avec des programmes dotés des fonctionnalités `grep`, `more`, `less`, `diff` et `cmp`.

Commandes utilisées dans les exercices :

### **bunzip2**

Décompresse un fichier compressé en `bzip2`.

### **bzcat**

Affiche le contenu d'un fichier compressé au format `bzip`.

### **bzip2**

Comprime les fichiers en utilisant l'algorithme et le format `bzip2`.

### **gunzip**

Décompresse un fichier compressé au format `gzip`.

### **gzip**

Comprime les fichiers en utilisant l'algorithme et le format `gzip`.

### **tar**

Crée, met à jour, répertorie et extrait des archives `tar`.

**unxz**

Décompresse un fichier compressé en xz.

**unzip**

Décompresse et extrait le contenu d'un fichier ZIP.

**xz**

Comprime les fichiers en utilisant l'algorithme et le format xz.

**zcat**

Affiche le contenu d'un fichier compressé au format gzip.

**zip**

Crée et compresse des archives ZIP.

## Réponses aux Exercices Guidés

1. D'après les extensions, lesquels des outils suivants ont été utilisés pour créer ces fichiers ?

Nom du fichier	tar	gzip	bzip2	xz
archive.tar	X			
archive.tgz	X	X		
archive.tar.xz	X			X

2. D'après les extensions, lesquels de ces fichiers sont des archives et lesquels sont compressés ?

Nom du fichier	Archive	Compressé
file.tar	X	
file.tar.bz2	X	X
file.zip	X	X
file.xz		X

3. Comment ajouter un fichier à une archive `tar` compressée au format `gzip` ?

Vous décompressez le fichier avec `gunzip`, vous ajoutez le fichier avec `tar uf`, puis vous le compressez avec `gzip`

4. Quelle option de `tar` demande à une archive `tar` d'inclure les chemins absolus commençant par `/` ?

L'option `-P`. De la page de manuel :

```
-P, --absolute-names
    Don't strip leading slashes from file names when creating archives
```

5. L'outil `zip` supporte-t-il différents niveaux de compression ?

Oui. Vous utiliseriez `-#`, en remplaçant `#` par un chiffre de 0 à 9. De la page de manuel :

```
-#
(-0, -1, -2, -3, -4, -5, -6, -7, -8, -9)
    Regulate the speed of compression using the specified digit #,
    where -0 indicates no compression (store all files), -1 indi-
```

cates the fastest compression speed (less compression) and -9 indicates the slowest compression speed (optimal compression, ignores the suffix list). The default compression level is -6.

Though still being worked, the intention is this setting will control compression speed for all compression methods. Currently only deflation is controlled.

## Réponses aux Exercices d'Exploration

1. Lors de l'extraction de fichiers, est-ce que `tar` supporte les globs dans la liste des fichiers ?

Oui, vous utiliseriez l'option `--wildcards`. L'option `--wildcards` doit être placé juste après le nom du fichier `tar` lorsque vous utilisez les options sans tiret. Par exemple :

```
$ tar xf tarfile.tar --wildcards dir/file*
$ tar --wildcards -xf tarfile.tar dir/file*
```

2. Comment s'assurer qu'un fichier décompressé est identique au fichier avant qu'il ne soit compressé ?

Vous n'avez pas besoin de faire quoi que ce soit avec les outils couverts dans cette leçon. Tous trois incluent des sommes de contrôle dans leur format de fichier qui est vérifiée lors de leur décompression.

3. Que se passe-t-il si vous essayez d'extraire un fichier d'une archive `tar` qui existe déjà sur votre système de fichiers ?

Le fichier sur votre système de fichiers est écrasé par la version qui se trouve dans le fichier `tar`.

4. Comment extraire le fichier `archive.tgz` sans utiliser l'option `tar z` ?

Vous le décompresserez d'abord avec `gunzip`.

```
$ gunzip archive.tgz
$ tar xf archive.tar
```



## 3.2 Recherche et extraction de données à partir de fichiers

### Référence aux objectifs de LPI

[Linux Essentials version 1.6, Exam 010, Objective 3.2](#)

### Valeur

3

### Domaines de connaissance les plus importants

- Tuyaux en ligne de commande
- Redirection d'entrée-sortie
- Expressions rationnelles élémentaires utilisant `.`, `[ ]`, `*`, and `?`

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `grep`
- `less`
- `cat`, `head`, `tail`
- `sort`
- `cut`
- `wc`





## 3.2 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	3 La Puissance de la Ligne de Commande
<b>Objectif :</b>	3.2 Recherche et Extraction de Données dans des Fichiers
<b>Leçon :</b>	1 sur 2

### Introduction

Dans cet atelier, nous nous concentrerons sur la redirection ou la transmission d'informations d'une source à une autre à l'aide d'outils spécifiques. La ligne de commande Linux redirige les informations par des canaux standard spécifiques. L'entrée standard (*stdin* ou canal 0) de la commande est considérée comme étant le clavier et la sortie standard (*stdout* ou canal 1) est considérée comme étant l'écran. Il existe également un autre canal qui est censé rediriger la sortie d'erreur (*stderr* ou canal 2) d'une commande ou les messages d'erreur d'un programme. L'entrée et/ou la sortie peuvent être redirigées.

Lors de l'exécution d'une commande, nous souhaitons parfois transmettre certaines informations à la commande ou rediriger la sortie vers un fichier spécifique. Chacune de ces fonctionnalités sera abordée dans les deux sections suivantes.

### Redirection des E/S

La redirection E/S (Entrée/Sortie) permet à l'utilisateur de rediriger des informations depuis ou vers une commande en utilisant un fichier texte. Comme décrit précédemment, l'entrée, la sortie

et la sortie d'erreur standard peuvent être redirigées, et les informations peuvent être extraites de fichiers texte.

## Redirection de la Sortie Standard

Pour rediriger la sortie standard vers un fichier, au lieu de l'écran, nous devons utiliser l'opérateur `>` suivi du nom du fichier. Si le fichier n'existe pas, un nouveau fichier sera créé, sinon, les informations écraseront le fichier existant.

Afin de voir le contenu du fichier que nous venons de créer, nous pouvons utiliser la commande `cat`. Par défaut, cette commande affiche le contenu d'un fichier à l'écran. Consultez la page du manuel pour en savoir plus sur ses fonctionnalités.

L'exemple ci-dessous démontre la fonctionnalité de l'opérateur. Dans un premier temps, un nouveau fichier est créé contenant le texte "Hello World!"

```
$ echo "Hello World!" > text
$ cat text
Hello World!
```

Dans la seconde invocation, le même fichier est écrasé par le nouveau texte :

```
$ echo "Hello!" > text
$ cat text
Hello!
```

Si nous voulons ajouter de nouvelles informations à la fin du fichier, nous utiliserons l'opérateur `>>`. Cet opérateur crée également un nouveau fichier s'il ne trouve pas un fichier existant.

Le premier exemple montre l'ajout du texte. Comme on peut le voir, le nouveau texte a été ajouté sur la ligne suivante :

```
$ echo "Hello to you too!" >> text
$ cat text
Hello!
Hello to you too!
```

Le deuxième exemple montre qu'un nouveau fichier sera créé :

```
$ echo "Hello to you too!" >> text2
```

```
$ cat text2
Hello to you too!
```

## Redirection de la Sortie d'Erreur Standard

Afin de rediriger uniquement les messages d'erreur, l'utilisateur devra utiliser l'opérateur `2>` suivi du nom du fichier dans lequel les erreurs seront écrites. Si le fichier n'existe pas, un nouveau fichier sera créé, sinon le fichier sera écrasé.

Comme expliqué, le canal pour rediriger l'erreur standard est le *canal 2*. Lors de la redirection de l'erreur standard, le canal doit être spécifié, contrairement à l'autre sortie standard où le *canal 1* est défini par défaut. Par exemple, la commande suivante recherche un fichier ou un répertoire nommé `games` et n'écrit que l'erreur dans le fichier `text-error`, tout en affichant la sortie standard à l'écran :

```
$ find /usr games 2> text-error
/usr
/usr/share
/usr/share/misc
-----Sortie Omise-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
$ cat text-error
find: `games': No such file or directory
```

**NOTE** Pour plus d'informations sur la commande `find`, consultez sa page man.

Par exemple, la commande suivante s'exécutera sans erreur, donc aucune information ne sera écrite dans le fichier `text-error` :

```
$ sort /etc/passwd 2> text-error
$ cat text-error
```

De même que la sortie standard, l'erreur standard peut également être ajoutée à un fichier avec l'opérateur `2>>`. La nouvelle erreur sera ainsi ajoutée à la fin du fichier. Si le fichier n'existe pas, un nouveau fichier sera créé. Le premier exemple montre l'ajout de la nouvelle information dans le fichier, tandis que le second exemple montre que la commande crée un nouveau fichier lorsqu'un fichier existant ne peut être trouvé avec le même nom :

```
$ sort /etc 2>> text-error
$ cat text-error
sort: read failed: /etc: Is a directory
```

```
$ sort /etc/shadow 2>> text-error2
$ cat text-error2
sort: open failed: /etc/shadow: Permission denied
```

En utilisant ce type de redirection, seuls les messages d'erreur seront redirigés vers le fichier, la sortie normale sera écrite à l'écran ou passera par la sortie standard ou *stdout*.

Il existe un fichier particulier qui, techniquement, est un "*bit bucket*" (un fichier qui accepte des données et n'en fait rien) : `/dev/null`. Vous pouvez rediriger toute information non pertinente que vous ne souhaitez pas voir affichée ou redirigée vers un fichier important, comme le montre l'exemple ci-dessous :

```
$ sort /etc 2> /dev/null
```

## Redirection de l'Entrée Standard

Ce type de redirection est utilisé pour entrer des données dans une commande, à partir d'un fichier spécifié au lieu d'un clavier. Dans ce cas, l'opérateur `<` est utilisé comme indiqué dans l'exemple :

```
$ cat < text
Hello!
Hello to you too!
```

La redirection de l'entrée standard est généralement utilisée avec les commandes qui n'acceptent pas les arguments de fichier. La commande `tr` est l'une d'entre elles. Cette commande peut être utilisée pour traduire le contenu d'un fichier en modifiant les caractères d'un fichier de manière spécifique, comme en supprimant un caractère particulier d'un fichier, l'exemple ci-dessous montre la suppression du caractère `l` :

```
$ tr -d "l" < text
Heo!
Heo to you too!
```

Pour plus d'informations, consultez la page man de `tr`.

## Here Documents (Ici documents)

Contrairement aux redirections de sortie, l'opérateur `<<` agit de manière différente par rapport aux autres opérateurs. Ce flux d'entrée est également appelé *here document*. *here document* représente le bloc de code ou de texte qui peut être redirigé vers la commande ou le programme interactif. Différents types de langages de script, comme `bash`, `sh` et `csh`, sont capables de prendre des données directement à partir de la ligne de commande, sans utiliser de fichiers texte.

Comme on peut le voir dans l'exemple ci-dessous, l'opérateur est utilisé pour entrer des données dans la commande, tandis que le mot après ne précise pas le nom du fichier. Le mot est interprété comme le délimiteur de l'entrée et il ne sera pas pris en considération comme contenu, donc `cat` ne l'affichera pas :

```
$ cat << hello
> hey
> ola
> hello
hey
ola
```

Consultez la page de manuel de la commande `cat` pour plus d'informations.

## Les Combinaisons

La première combinaison que nous allons explorer combine la redirection de la sortie standard et de la sortie d'erreur standard vers le même fichier. Les opérateurs `&>` et `&>>` sont utilisés, `&` représentant la combinaison du *canal 1* et du *canal 2*. Le premier opérateur écrasera le contenu existant du fichier et le second ajoutera les nouvelles informations à la fin du fichier. Les deux opérateurs permettront la création du nouveau fichier s'il n'existe pas, tout comme dans les sections précédentes :

```
$ find /usr admin &> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
-----Sortie omise-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
```

```

/usr/games
find: `admin': No such file or directory
$ find /etc/calendar &>> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
-----Sortie omise-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
/etc/calendar
/etc/calendar/default

```

Examinons un exemple utilisant la commande `cut` :

```

$ cut -f 3 -d "/" newfile
$ cat newfile

share
share
share
-----Sortie omise-----
lib
games
find: `admin': No such file or directory
calendar
calendar
find: `admin': No such file or directory

```

La commande `cut` coupe des champs spécifiés du fichier d'entrée en utilisant l'option `-f`, le 3ème champ dans notre cas. Pour que la commande puisse trouver le champ, un délimiteur doit également être spécifié avec l'option `-d`. Dans notre cas, le délimiteur sera le caractère `/`.

Pour en savoir plus sur la commande de `cut`, consultez sa page man.

## Les Pipes de la Ligne de Commande

La redirection est principalement utilisée pour stocker le résultat d'une commande, pour être traité par une commande différente. Ce type de processus intermédiaire peut devenir très fastidieux et compliqué si vous voulez que les données passent par plusieurs processus. Pour

éviter cela, vous pouvez relier la commande directement par des *pipes* (tuyaux). En d'autres termes, la sortie de la première commande devient automatiquement l'entrée de la deuxième commande. Cette connexion se fait à l'aide de l'opérateur `|` (barre verticale) :

```
$ cat /etc/passwd | less
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
:
```

Dans l'exemple ci-dessus, la commande `less` après l'opérateur de pipe modifie la façon dont le fichier est affiché. La commande `less` affiche le fichier texte, ce qui permet à l'utilisateur de faire défiler une ligne à la fois. `less` est également utilisé par défaut pour afficher les pages de manuel, comme nous l'avons vu dans les leçons précédentes.

Il est possible d'utiliser plusieurs pipes en même temps. Les commandes intermédiaires qui reçoivent l'entrée puis la modifient et produisent la sortie sont appelées *filtres*. Prenons la commande `ls -l` et essayons de compter le nombre de mots des 10 premières lignes de la sortie. Pour ce faire, nous devons utiliser la commande `head` qui, par défaut, affiche les 10 premières lignes d'un fichier, puis compter les mots à l'aide de la commande `wc` :

```
$ ls -l | head | wc -w
10
```

Comme mentionné précédemment, par défaut, `head` n'affiche que les 10 premières lignes du fichier texte spécifié. Ce comportement peut être modifié en utilisant des options spécifiques. Consultez la page de manuel de la commande pour en savoir plus.

Il existe une autre commande qui affiche la fin d'un fichier : `tail`. Par défaut, cette commande sélectionne les 10 dernières lignes et les affiche, comme `head`, ce nombre peut également être modifié. Consultez la page de manuel de `tail` pour plus de détails.

**NOTE**

L'option `-f` peut afficher les dernières lignes d'un fichier pendant qu'il est mis à jour. Cette fonction peut s'avérer très utile pour surveiller des fichiers comme `syslog` pour une activité en cours.

La commande `wc` (word count) compte par défaut les lignes, les mots et les octets d'un fichier. Comme le montre l'exercice, l'option `-w` fait en sorte que la commande ne compte que les mots contenus dans les lignes sélectionnées. Les options les plus courantes que vous pouvez utiliser avec cette commande sont les suivantes : `-l`, qui indique à la commande de ne compter que les

lignes, et `-c`, qui est utilisé pour ne compter que les octets. D'autres variantes et options de la commande, ainsi que de plus amples informations sur `wc` peuvent être trouvées dans la page de man de la commande.



## Exercices Guidés

1. Listez le contenu de votre répertoire courant, y compris la propriété et les autorisations, et redirigez la sortie vers un fichier nommé `contents.txt` dans votre répertoire personnel.

2. Triez le contenu du fichier `contents.txt` de votre répertoire courant et ajoutez-le à la fin d'un nouveau fichier nommé `contents-sorted.txt`.

3. Affichez les 10 dernières lignes du fichier `/etc/passwd` et redirigez-les vers un nouveau fichier dans le répertoire `Documents` de votre utilisateur.

4. Comptez le nombre de mots dans le fichier `contents.txt` et ajoutez la sortie à la fin d'un fichier `field2.txt` dans votre répertoire personnel. Vous devrez utiliser la redirection en entrée et en sortie.

5. Affichez les 5 premières lignes du fichier `/etc/passwd` et triez la sortie par ordre alphabétique inverse.

6. En utilisant le fichier `contents.txt` créé précédemment, comptez le nombre de caractères des 9 dernières lignes.

7. Comptez le nombre de fichiers appelés `test` dans le répertoire `/usr/share` et ses sous-répertoires. Note : chaque ligne de sortie de la commande `find` représente un fichier.

## Exercices d'Exploration

1. Sélectionnez le deuxième champ du fichier `contents.txt` et redirigez la sortie standard et la sortie d'erreur vers un autre fichier appelé `field1.txt`.

2. En utilisant l'opérateur de redirection d'entrée et la commande `tr`, supprimez les tirets (-) du fichier `contents.txt`.

3. Quel est le plus grand avantage de ne rediriger que les erreurs vers un fichier ?

4. Remplacez tous les espaces récurrents dans le fichier `contents.txt`, trié alphabétiquement, par un seul espace.

5. Dans une ligne de commande, éliminez les espaces récurrents (comme vous l'avez fait dans l'exercice précédent), sélectionnez le neuvième champ et triez-le par ordre alphabétique inverse et sans tenir compte de la casse. Combien de pipes avez-vous dû utiliser ?

## Résumé

Dans cet atelier, vous avez appris :

- Les types de redirection
- Comment utiliser les opérateurs de redirection
- Comment utiliser les pipes pour filtrer les sorties des commandes

Commandes utilisées dans cette leçon :

### **cut**

Supprime des sections de chaque ligne d'un fichier.

### **cat**

Affiche ou concatène des fichiers.

### **find**

Recherche de fichiers dans une hiérarchie de répertoires.

### **less**

Affiche un fichier, ce qui permet à l'utilisateur de faire défiler une ligne à la fois.

### **more**

Affiche un fichier, une page à la fois.

### **head**

Affiche les 10 premières lignes d'un fichier.

### **tail**

Affiche les 10 dernières lignes d'un fichier.

### **sort**

Trie les fichiers.

### **wc**

Compte par défaut les lignes, les mots ou les octets d'un fichier.

## Réponses aux Exercices Guidés

1. Listez le contenu de votre répertoire courant, y compris la propriété et les autorisations, et redirigez la sortie vers un fichier appelé `contents.txt` dans votre répertoire personnel.

```
$ ls -l > contents.txt
```

2. Triez le contenu du fichier `contents.txt` de votre répertoire courant et ajoutez-le à la fin d'un nouveau fichier nommé `contents-sorted.txt`.

```
$ sort contents.txt >> contents-sorted.txt
```

3. Affichez les 10 dernières lignes du fichier `/etc/passwd` et redirigez-les vers un nouveau fichier dans le répertoire `Documents` de votre utilisateur.

```
$ tail /etc/passwd > Documents/newfile
```

4. Comptez le nombre de mots dans le fichier `contents.txt` et ajoutez la sortie à la fin d'un fichier `field2.txt` dans votre répertoire personnel. Vous devrez utiliser la redirection en entrée et en sortie.

```
$ wc < contents.txt >> field2.txt
```

5. Affichez les 5 premières lignes du fichier `/etc/passwd` et triez la sortie par ordre alphabétique inverse.

```
$ head -n 5 /etc/passwd | sort -r
```

6. En utilisant le fichier `contents.txt` créé précédemment, comptez le nombre de caractères des 9 dernières lignes.

```
$ tail -n 9 contents.txt | wc -c  
531
```

7. Comptez le nombre de fichiers appelés `test` dans le répertoire `/usr/share` et ses sous-répertoires. Note : chaque ligne de sortie de la commande `find` représente un fichier.

```
$ find /usr/share -name test | wc -l  
125
```

## Réponses aux Exercices d'Exploration

1. Sélectionnez le deuxième champ du fichier `contents.txt` et redirigez la sortie standard et la sortie d'erreur vers un autre fichier appelé `field1.txt`.

```
$ cut -f 2 -d " " contents.txt &> field1.txt
```

2. En utilisant l'opérateur de redirection d'entrée et la commande `tr`, supprimez les tirets (-) du fichier `contents.txt`.

```
$ tr -d "-" < contents.txt
```

3. Quel est le plus grand avantage de ne rediriger que les erreurs vers un fichier ?

Seul le fait de rediriger les erreurs vers un fichier peut aider à conserver un fichier journal qui est fréquemment surveillé.

4. Remplacez tous les espaces récurrents dans le fichier `contents.txt`, trié alphabétiquement, par un seul espace.

```
$ sort contents.txt | tr -s " "
```

5. Dans une ligne de commande, éliminez les espaces récurrents (comme vous l'avez fait dans l'exercice précédent), sélectionnez le neuvième champ et triez-le par ordre alphabétique inverse et sans tenir compte de la casse. Combien de pipes avez-vous dû utiliser ?

```
$ cat contents.txt | tr -s " " | cut -f 9 -d " " | sort -fr
```

L'exercice utilise 3 pipes, un pour chaque filtre.



## 3.2 Leçon 2

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	3 La Puissance de la Ligne de Commande
<b>Objectif :</b>	3.2 Recherche et Extraction de données dans des Fichiers
<b>Leçon :</b>	2 sur 2

### Introduction

Dans cette leçon, nous allons examiner les outils qui sont utilisés pour manipuler le texte. Ces outils sont fréquemment utilisés par les administrateurs système ou les programmes pour surveiller ou identifier automatiquement des informations spécifiques récurrentes.

### Recherche dans les Fichiers avec `grep`

Le premier outil dont nous parlerons dans cette leçon est la commande `grep`. `grep` est l'abréviation de l'expression "global regular expression print" et sa principale fonctionnalité est de rechercher dans des fichiers un motif spécifié. La commande affiche les lignes contenant le motif spécifié, identifié en rouge.

```
$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1001:1001:User,,,:/home/user:/bin/bash
```

`grep`, comme la plupart des commandes, peut également être modifiée en utilisant des options. Voici les plus courantes :

**-i**

la recherche est insensible à la casse

**-r**

la recherche est réursive (elle recherche dans tous les fichiers du répertoire donné et de ses sous-répertoires)

**-c**

la recherche compte le nombre de résultats

**-v**

inverser la correspondance, pour afficher les lignes qui ne correspondent pas au terme recherché

**-E**

active les expressions régulières étendues (nécessaires à certains méta-caractères plus avancés comme `|`, `+` et `?`)

La commande `grep` dispose de nombreuses autres options utiles. Consultez sa page de manuel pour en savoir plus.

## Les Expressions Régulières

Le deuxième outil est très puissant. Il est utilisé pour décrire des morceaux de texte dans les fichiers, également appelés *expressions régulières*. Les expressions régulières sont extrêmement utiles pour extraire des données de fichiers texte en construisant des motifs. Elles sont couramment utilisées dans les scripts ou lors de la programmation avec des langages de haut niveau, tels que Perl ou Python.

Lorsque vous travaillez avec des expressions régulières, il est très important de garder à l'esprit que *chaque caractère compte* et que le motif est écrit dans le but de faire correspondre une séquence spécifique de caractères, appelée chaîne de caractères. La plupart des motifs utilisent les symboles ASCII normaux, tels que les lettres, les chiffres, la ponctuation ou d'autres symboles, mais ils peuvent également utiliser des caractères Unicode afin de correspondre à tout autre type de texte.

La liste suivante explique les méta-caractères d'expressions régulières qui sont utilisés pour former les motifs.



.

Correspond à un seul caractère (sauf nouvelle ligne)

**[abcABC]**

Fait correspondre un des caractères entre crochets

**[^abcABC]**

Fait correspondre n'importe quel caractère sauf ceux entre crochets

**[a-z]**

Correspond à n'importe quel caractère de la plage

**[^a-z]**

Correspond à n'importe quel caractère sauf ceux de la plage

**soleil|lune**

Trouve l'une ou l'autre des chaînes énumérées

**^**

Début d'une ligne

**\$**

Fin de ligne

Toutes les fonctionnalités des expressions régulières peuvent également être implémentées avec **grep**. Vous pouvez voir que dans l'exemple ci-dessus, le mot n'est pas entouré de guillemets. Pour éviter que le shell n'interprète le méta-caractère lui-même, il est recommandé que le motif le plus complexe se trouve entre guillemets doubles (" "). Pour des raisons pratiques, nous utiliserons des guillemets doubles lors de la mise en œuvre d'expressions régulières. Les autres guillemets conservent leur fonctionnalité normale, comme nous l'avons vu dans les leçons précédentes.

Les exemples suivants soulignent la fonctionnalité des expressions régulières. Nous aurons besoin de données dans le fichier, c'est pourquoi la prochaine série de commandes ne fait qu'ajouter différentes chaînes de caractères au fichier `text.txt`.

```
$ echo "aaabbb1" > text.txt
$ echo "abab2" >> text.txt
$ echo "noone2" >> text.txt
$ echo "class1" >> text.txt
$ echo "alien2" >> text.txt
$ cat text.txt
```

```
aaabbb1
abab2
noone2
class1
alien2
```

Le premier exemple est une combinaison de recherche dans le fichier sans et avec des expressions régulières. Afin de bien comprendre les expressions régulières, il est très important de montrer la différence. La première commande recherche la chaîne de caractères exacte, n'importe où sur la ligne, tandis que la deuxième commande recherche les ensembles de caractères qui contiennent l'un des caractères entre crochets. Par conséquent, les résultats des commandes sont différents.

```
$ grep "ab" text.txt
aaabbb1
abab2
$ grep "[ab]" text.txt
aaabbb1
abab2
class1
alien2
```

La deuxième série d'exemples montre l'application du méta-caractère de début et de fin de ligne. Il est très important de préciser la nécessité de placer les 2 caractères au bon endroit dans l'expression. Lorsque vous spécifiez le début de la ligne, le méta-caractère doit être avant l'expression, tandis que lorsque vous spécifiez la fin de la ligne, le méta-caractère doit être après l'expression.

```
$ grep "^a" text.txt
aaabbb1
abab2
alien2
$ grep "2$" text.txt
abab2
noone2
alien2
```

En plus des méta-caractères expliqués précédemment, les expressions régulières ont également des méta-caractères qui permettent la multiplication du motif spécifié précédemment :

**\***

Zéro ou plusieurs occurrences du motif précédent

**+**

Une ou plusieurs occurrences du motif précédent

**?**

Zéro ou une occurrence du motif précédent

Pour les méta-caractères multiplicateurs, la commande ci-dessous recherche une chaîne qui contient `ab`, un seul caractère et un ou plusieurs des caractères précédemment trouvés. Le résultat montre que `grep` a trouvé la chaîne de caractères `aaabbb1`, correspondant à la partie `abbb` ainsi qu'à `abab2`. Comme le caractère `+` est un caractère d'expression régulière *étendue*, nous devons passer l'option `-E` à la commande `grep`.

```
$ grep -E "ab.+" text.txt
aaabbb1
abab2
```

La plupart des méta-caractères sont explicites, mais ils peuvent devenir délicats à la première utilisation. Les exemples précédents représentent une petite partie de la fonctionnalité des expressions régulières. Essayez tous les méta-caractères du tableau ci-dessus pour mieux comprendre leur fonctionnement.

## Exercices Guidés

En utilisant `grep` et le fichier `/usr/share/hunspell/en_US.dic`, trouvez les lignes qui correspondent aux critères suivants :

1. Toutes les lignes contenant le mot `cat` n'importe où sur la ligne.

2. Toutes les lignes qui ne contiennent aucun des caractères suivants : `sawgtfixk`.

3. Toutes les lignes qui commencent par 3 lettres quelconques suivi de la chaîne `dig`.

4. Toutes les lignes qui se terminent par au moins un `e`.

5. Toutes les lignes qui contiennent l'un des mots suivants : `org`, `kay` ou `tuna`.

6. Le nombre de lignes qui commencent, ou non, par un `c` suivi de la chaîne `ati`.

## Exercices d'Exploration

1. Trouvez l'expression régulière qui correspond aux mots de la ligne "Inclure" et qui ne correspond pas à ceux de la ligne "Exclure" :

◦ Inclure : pot, spot, apot

Exclure : potic, spots, potatoe

◦ Inclure : arp99, apple, zipper

Exclure : zoo, arive, attack

◦ Inclure : arcane, capper, zoology

Exclure : air, coper, zoloc

◦ Inclure : 0th/pt, 3th/tc, 9th/pt

Exclure : 0/nm, 3/nm, 9/nm

◦ Inclure : Hawaii, Dario, Ramiro

Exclure : hawaii, Ian, Alice

2. Quelle autre commande utile est couramment utilisée pour rechercher dans les fichiers ? Quelles sont les fonctionnalités supplémentaires dont elle dispose ?
3. En repensant à la leçon précédente, utilisez un des exemples et essayez de rechercher un motif spécifique dans la sortie de la commande, avec l'aide de `grep`.

## Résumé

Dans cet atelier, vous avez appris :

- Les méta-caractères des expressions régulières
- Comment créer des motifs avec des expressions régulières
- Comment rechercher dans les fichiers

Commandes utilisées dans les exercices :

### **grep**

Recherche un caractère ou une chaîne de caractères dans un fichier

## Réponses aux Exercices Guidés

En utilisant `grep` et le fichier `/usr/share/hunspell/en_US.dic`, trouvez les lignes qui correspondent aux critères suivants :

1. Toutes les lignes contenant le mot `cat` n'importe où sur la ligne.

```
$ grep "cat" /usr/share/hunspell/en_US.dic
Alcatraz/M
Decatur/M
Hecate/M
...
```

2. Toutes les lignes qui ne contiennent aucun des caractères suivants : `sawgtfixk`.

```
$ grep -v "[sawgtfixk]" /usr/share/hunspell/en_US.dic
49269
0/nm
1/n1
2/nm
2nd/p
3/nm
3rd/p
4/nm
5/nm
6/nm
7/nm
8/nm
...
```

3. Toutes les lignes qui commencent par 3 lettres quelconques suivi de la chaîne `dig`.

```
$ grep "^...dig" /usr/share/hunspell/en_US.dic
cardigan/SM
condign
predigest/GDS
...
```

4. Toutes les lignes qui se terminent par au moins un `e`.

```
$ grep -E "e+$" /usr/share/hunspell/en_US.dic
Anglicize
Anglophobe
Anthropocene
...
```

5. Toutes les lignes qui contiennent l'un des mots suivants : org, kay ou tuna.

```
$ grep -E "org|kay|tuna" /usr/share/hunspell/en_US.dic
Borg/SM
George/MS
Tokay/M
fortunate/UY
...
```

6. Le nombre de lignes qui commencent, ou non, par un c suivi de la chaîne ati.

```
$ grep -cE "^c?ati" /usr/share/hunspell/en_US.dic
3
```



## Réponses aux Exercices d'Exploration

1. Trouvez l'expression régulière qui correspond aux mots de la ligne "Inclure" et qui ne correspond pas à ceux de la ligne "Exclure" :

◦ Inclure : pot, spot, apot

Exclure : potic, spots, potatoe

Réponse : pot\$

◦ Inclure : arp99, apple, zipper

Exclure : zoo, arive, attack

Réponse : p+

◦ Inclure : arcane, capper, zoology

Exclure : air, coper, zoloc

Réponse : arc|cap|zoo

◦ Inclure : 0th/pt, 3th/tc, 9th/pt

Exclure : 0/nm, 3/nm, 9/nm

Réponse : [0-9]th. +

◦ Inclure : Hawaii, Dario, Ramiro

Exclure : hawaii, Ian, Alice

Réponse : ^[A-Z]a.\*i+

2. Quelle autre commande utile est couramment utilisée pour rechercher dans les fichiers ? Quelles sont les fonctionnalités supplémentaires dont elle dispose ?

La commande `sed`. Cette commande permet de trouver et de remplacer des caractères ou des ensembles de caractères dans un fichier.

3. En repensant à la leçon précédente, utilisez un des exemples et essayez de rechercher un motif spécifique dans la sortie de la commande, avec l'aide de `grep`.

J'ai pris une des réponses des exercices d'exploration et j'ai cherché la ligne qui a comme

autorisation de groupe : lecture, écriture et exécution. Votre réponse peut être différente, selon la commande que vous avez choisie et le motif que vous avez créé.

```
$ cat contents.txt | tr -s " " | grep "^...rwx"
```

Cet exercice a pour but de vous montrer que `grep` peut également recevoir des entrées de différentes commandes et qu'il peut aider à filtrer les informations générées.



## 3.3 Conversion de commandes en script

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 3.3

### Valeur

4

### Domaines de connaissance les plus importants

- Faire des scripts shell simples
- Connaître des éditeurs de texte courants (vi et nano)

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `#!` (shebang)
- `/bin/bash`
- Variables
- Arguments
- boucle `for`
- `echo`
- Statut de sortie



## 3.3 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	3 La Puissance de la Ligne de Commande
<b>Objectif :</b>	3.3 Transformer les Commandes en Script
<b>Leçon :</b>	1 sur 2

### Introduction

Jusqu'à présent, nous avons appris à exécuter des commandes depuis le shell, mais nous pouvons également entrer des commandes dans un fichier, puis configurer ce fichier pour qu'il soit exécutable. Lorsque le fichier est exécuté, ces commandes sont exécutées l'une après l'autre. Ces fichiers exécutables sont appelés *scripts*, et ils constituent un outil absolument crucial pour tout administrateur système Linux. En gros, on peut considérer que Bash est un langage de programmation aussi bien qu'un shell.

### Affichage de la Sortie

Commençons par démontrer une commande que vous avez peut-être déjà vue dans les leçons précédentes : `echo` affichera un argument sur la sortie standard.

```
$ echo "Hello World!"  
Hello World!
```

Maintenant, nous allons utiliser la redirection de fichier pour envoyer cette commande à un

nouveau fichier appelé `new_script`.

```
$ echo 'echo "Hello World!"' > new_script
$ cat new_script
echo "Hello World!"
```

Le fichier `new_script` contient maintenant la même commande précédente.

## Rendre un Script Exécutable

Montrons quelques-unes des étapes nécessaires pour que ce fichier s'exécute comme nous le souhaitons. La première idée d'un utilisateur pourrait être de simplement taper le nom du script, comme il pourrait taper le nom de n'importe quelle autre commande :

```
$ new_script
/bin/bash: new_script: command not found
```

Nous supposons sans risque que `new_script` existe dans notre emplacement actuel, mais remarquez que le message d'erreur ne nous dit pas que le *fichier* n'existe pas, il nous dit que la *commande* n'existe pas. Il serait utile de discuter de la manière dont Linux gère les commandes et les exécutables.

## Les Commandes et PATH

Lorsque nous tapons la commande `ls` dans le shell, par exemple, nous exécutons un fichier appelé `ls` qui existe dans notre système de fichiers. Vous pouvez le prouver en utilisant la commande `which` :

```
$ which ls
/bin/ls
```

Il deviendrait rapidement fastidieux de taper le chemin absolu de `ls` chaque fois que l'on souhaite consulter le contenu d'un répertoire, c'est pourquoi Bash dispose d'une *variable d'environnement* qui contient tous les répertoires où l'on peut trouver les commandes que l'on souhaite exécuter. Vous pouvez consulter le contenu de cette variable en utilisant `echo`.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
```

```
ap/bin
```

Dans chacun de ces emplacements le shell s'attend à trouver la commande, délimitée par des deux-points (:). Vous remarquerez que `/bin` est présent, mais on peut supposer sans risque que notre emplacement actuel ne l'est pas. Le shell cherchera `new_script` dans chacun de ces répertoires, mais il ne le trouvera pas et enverra donc l'erreur que nous avons vue ci-dessus.

Il y a trois solutions à ce problème : nous pouvons déplacer `new_script` dans l'un des répertoires de `PATH`, nous pouvons ajouter notre répertoire actuel à `PATH`, ou nous pouvons changer la façon dont nous essayons d'appeler le script. Cette dernière solution est la plus simple, elle nous oblige simplement à spécifier l'emplacement actuel lors de l'appel du script en utilisant un point suivi de la barre oblique (`./`).

```
$ ./new_script
/bin/bash: ./new_script: Permission denied
```

Le message d'erreur a changé, ce qui indique que nous avons fait quelques progrès.

## Permission d'Exécution

La première investigation qu'un utilisateur doit faire dans ce cas est d'utiliser `ls -l` pour examiner le fichier :

```
$ ls -l new_script
-rw-rw-r-- 1 user user 20 Apr 30 12:12 new_script
```

Nous pouvons voir que les autorisations pour ce fichier sont fixées à 664 par défaut. Nous n'avons pas encore défini les *permissions d'exécution* pour ce fichier.

```
$ chmod +x new_script
$ ls -l new_script
-rwxrwxr-x 1 user user 20 Apr 30 12:12 new_script
```

Cette commande a donné des autorisations d'exécution à *tous* les utilisateurs. Sachez que cela peut constituer un risque pour la sécurité, mais pour l'instant, il s'agit d'un niveau de permission acceptable.

```
$ ./new_script
```

```
Hello World!
```

Nous sommes maintenant en mesure d'exécuter notre script.

## Définir l'Interprète

Comme nous l'avons démontré, nous avons pu simplement entrer du texte dans un fichier, le définir comme un exécutable et l'exécuter. `new_script` est fonctionnellement toujours un fichier texte normal, mais nous avons réussi à le faire interpréter par Bash. Mais qu'en est-il s'il est écrit en Perl, ou en Python ?

C'est une très bonne pratique de spécifier le type d'interprète que nous voulons utiliser dans la première ligne d'un script. Cette ligne est appelée une *ligne bang* ou plus communément un *shebang*. Elle indique au système comment nous voulons que ce fichier soit exécuté. Puisque nous apprenons le Bash, nous utiliserons le chemin absolu de notre exécutable Bash, en utilisant encore une fois `which` :

```
$ which bash
/bin/bash
```

Notre shebang commence par un dièse et un point d'exclamation, suivi du chemin absolu ci-dessus. Ouvrons `new_script` dans un éditeur de texte et insérons le shebang. Profitons également de l'occasion pour insérer un *commentaire* dans notre script. Les commentaires sont ignorés par l'interprète. Ils sont écrits pour le bénéfice des autres utilisateurs qui souhaitent comprendre votre script.

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de documenter tous les
scripts.

echo "Hello World!"
```

Nous allons également apporter une modification supplémentaire au nom du fichier : nous allons enregistrer ce fichier sous le nom `new_script.sh`. Le suffixe `".sh"` ne modifie en rien l'exécution du fichier. Par convention, les scripts bash prennent l'extension `.sh` ou `.bash` afin de les identifier plus facilement, de la même manière que les scripts Python sont généralement identifiés avec le suffixe `.py`.

## Les Éditeurs de Textes Populaires

Les utilisateurs de Linux doivent souvent travailler dans un environnement où les éditeurs de texte graphiques ne sont pas disponibles. Il est donc fortement recommandé de se familiariser au moins un peu avec l'édition de fichiers texte à partir de la ligne de commande. Deux des éditeurs de texte les plus courants sont `vi` et `nano`.

### vi

`vi` est un éditeur de texte vénérable et est installé par défaut sur presque tous les systèmes Linux existants. `vi` a donné naissance à un clone appelé `vi IMproved` ou `vim` qui ajoute certaines fonctionnalités mais maintient l'interface de `vi`. Bien que travailler avec `vi` soit intimidant pour un nouvel utilisateur, l'éditeur est populaire et très apprécié des utilisateurs qui apprennent ses nombreuses fonctionnalités.

La différence la plus importante entre `vi` et les applications telles que Notepad est que `vi` a trois modes différents. Au démarrage, les touches `H`, `J`, `K` et `L` sont utilisées pour naviguer, et non pour taper. Dans ce *mode navigation*, vous pouvez appuyer sur `I` pour entrer en *mode insertion*. À ce stade, vous pouvez taper normalement. Pour quitter le *mode insertion*, vous appuyez sur `Echap` pour revenir au *mode navigation*. Dans le *mode navigation*, vous pouvez appuyer sur `:` pour entrer en *mode commande*. À partir de ce mode, vous pouvez sauvegarder, supprimer, quitter ou modifier des options.

Bien que `vi` ait une longue courbe d'apprentissage, les différents modes peuvent, avec le temps, permettre à un utilisateur expérimenté de devenir plus efficace qu'avec d'autres éditeurs.

### nano

`nano` est un outil plus récent, conçu pour être simple et plus facile à utiliser que `vi`. `nano` n'a pas de modes différents. Au lieu de cela, un utilisateur au démarrage peut commencer à taper, et utiliser la touche `Ctrl` pour accéder aux outils affichés au bas de l'écran.

```
[ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Get Help    ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit        ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo
```

Les éditeurs de texte sont une question de préférence personnelle, et l'éditeur que vous choisissez d'utiliser n'aura aucune incidence sur cette leçon. Mais se familiariser avec un ou plusieurs éditeurs de texte sera payant à l'avenir.



## Les Variables

Les variables sont une partie importante de tout langage de programmation, et Bash n'est pas différent. Lorsque vous démarrez une nouvelle session à partir du terminal, le shell définit déjà certaines variables pour vous. La variable `PATH` en est un exemple. Nous les appelons *variables d'environnement*, car elles définissent généralement les caractéristiques de l'environnement de notre shell. Vous pouvez modifier et ajouter des variables d'environnement, mais pour l'instant, concentrons-nous sur la définition des variables à l'intérieur de notre script.

Nous allons modifier notre script pour qu'il ressemble à ceci :

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les scripts.

username=Carol

echo "Hello $username!"
```

Dans ce cas, nous avons créé une *variable* appelée `username` et nous lui avons attribué la *valeur* `Carol`. Veuillez noter qu'il n'y a pas d'espace entre le nom de la variable, le signe égal, et la valeur attribuée.

Dans la ligne suivante, nous avons utilisé la commande `echo` avec la variable, mais il y a un signe de dollar (\$) devant le nom de la variable. C'est important, car il indique au shell que nous souhaitons traiter `username` comme une variable, et pas seulement comme un mot normal. En entrant `$username` dans notre commande, nous indiquons que nous voulons effectuer une *substitution*, en remplaçant le *nom* d'une variable par la *valeur* attribuée à cette variable.

En exécutant le nouveau script, nous obtenons cette sortie :

```
$ ./new_script.sh
Hello Carol!
```

- Les variables ne doivent contenir que des caractères alphanumériques ou des traits de soulignement (underscores), et sont sensibles à la casse. `Username` et `username` seront traitées comme des variables distinctes.
- La substitution de variable peut également avoir le format `${username}`, avec l'ajout des `{ }`. Ceci est également acceptable.

- Les variables en Bash ont un *type implicite*, et sont considérées comme des chaînes de caractères. Cela signifie que l'exécution de fonctions mathématiques en Bash est plus compliquée que dans d'autres langages de programmation tels que C/C++ :

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username=Carol
x=2
y=4
z=$((x+y))
echo "Hello $username!"
echo "$x + $y"
echo "$z"
```

```
$ ./new_script.sh
Hello Carol!
2 + 4
2+4
```

## L'Utilisation des Guillemets avec des Variables

Modifions comme suit la valeur de notre variable `username` :

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username="Carol Smith"

echo "Hello $username!"
```

L'exécution de ce script nous donnera une erreur :

```
$ ./new_script.sh
./new_script.sh: line 5: Smith: command not found
```

```
Hello !
```

Gardez à l'esprit que Bash est un interprète et qu'en tant que tel, il *interprète* notre script ligne par ligne. Dans ce cas, il interprète correctement `username=Carol` pour définir une variable `username` avec la valeur `Carol`. Mais il interprète ensuite l'espace comme indiquant la fin de cette tâche, et `Smith` comme étant le nom d'une commande. Pour que l'espace et le nom `Smith` soient inclus comme nouvelle valeur de notre variable, nous mettrons des guillemets doubles (`"`) autour du nom.

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username="Carol Smith"

echo "Hello $username!"
```

```
$ ./new_script.sh
Hello Carol Smith!
```

Une chose importante à noter dans Bash est que les guillemets doubles et les guillemets simples (`'`) se comportent très différemment. Les guillemets doubles sont considérés comme “faibles”, car ils permettent à l'interprète d'effectuer des substitutions à l'intérieur des guillemets. Les guillemets simples sont considérés comme “forts”, parce qu'ils empêchent toute substitution de se produire. Prenons l'exemple suivant :

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username="Carol Smith"

echo "Hello $username!"
echo 'Hello $username!'
```

```
$ ./new_script.sh
Hello Carol Smith!
```

```
Hello $username!
```

Dans la deuxième commande `echo`, l'interprète a été empêché de remplacer `$username` par `Carol Smith`, et la sortie est donc prise littéralement.

## Les Arguments

Vous êtes déjà familiarisé avec l'utilisation d'arguments dans les utilitaires de base de Linux. Par exemple, `rm fichiertest` contient à la fois l'exécutable `rm` et un argument `fichiertest`. Les arguments peuvent être passés au script lors de l'exécution, et modifieront le comportement du script. Ils sont faciles à mettre en œuvre.

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username=$1

echo "Hello $username!"
```

Au lieu d'attribuer une valeur à `username` directement à l'intérieur du script, nous lui attribuons la valeur d'une nouvelle variable `$1`, qui correspond à la valeur du *premier argument*.

```
$ ./new_script.sh Carol
Hello Carol!
```

Les neuf premiers arguments sont traités de cette manière. Il existe des moyens de traiter plus de neuf arguments, mais cela dépasse le cadre de cette leçon. Nous allons montrer un exemple en utilisant seulement deux arguments :

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username1=$1
username2=$2
echo "Hello $username1 and $username2!"
```

```
$ ./new_script.sh Carol Dave
Hello Carol and Dave!
```

Il y a une considération importante à prendre en compte lorsque l'on utilise des arguments : Dans l'exemple ci-dessus, il y a deux arguments, `Carol` et `Dave`, attribués respectivement à `$1` et `$2`. Si le deuxième argument est manquant, par exemple, le shell ne produira pas d'erreur. La valeur de `$2` sera simplement *null*, ou rien du tout.

```
$ ./new_script.sh Carol
Hello Carol and !
```

Dans notre cas, il serait bon d'introduire une certaine logique dans notre script afin que différentes *conditions* affectent la *sortie* que nous souhaitons afficher. Nous commencerons par introduire une autre variable utile, puis nous passerons à la création des *déclarations if*.

## Retourner le Nombre d'Arguments

Alors que des variables telles que `$1` et `$2` contiennent la valeur des arguments de position, une autre variable `##` contient le *nombre d'arguments*.

```
#!/bin/bash

# C'est notre premier commentaire. C'est aussi une bonne pratique de commenter tous les
scripts.

username=$1

echo "Hello $username!"
echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol Dave
Hello Carol!
Number of arguments: 2.
```

## La Logique Conditionnelle

L'utilisation de la logique conditionnelle dans la programmation est un vaste sujet, qui ne sera pas traité en profondeur dans cette leçon. Nous nous concentrerons sur la *syntaxe* des conditions en

Bash, qui diffère de la plupart des autres langages de programmation.

Commençons par passer en revue ce que nous espérons réaliser. Nous disposons d'un script simple qui devrait afficher un message de salutation à un seul utilisateur. S'il y a autre chose qu'un seul utilisateur, nous devrions afficher un message d'erreur.

- La *condition* que nous testons est le nombre d'utilisateurs, qui est contenu dans la variable `$#`. Nous aimerions savoir si la valeur de `$#` est 1.
- Si la condition est *vraie*, l'*action* que nous ferons sera de saluer l'utilisateur.
- Si la condition est *fausse*, nous affichons un message d'erreur.

Maintenant que la logique est claire, nous allons nous concentrer sur la *syntaxe* requise pour mettre en œuvre cette logique.

```
#!/bin/bash

# Un script simple pour saluer un seul utilisateur.

if [ $# -eq 1 ]
then
    username=$1

    echo "Hello $username!"
else
    echo "Veuillez ne saisir qu'un seul argument."
fi
echo "Nombre d'arguments : $#."
```

La logique conditionnelle est contenue entre `if` et `fi`. La condition à tester est située entre crochets `[ ]`, et la mesure à prendre si la condition est vraie est indiquée après `then`. Notez les espaces entre les crochets et la logique contenue. L'omission de cet espace entraînera des erreurs.

Ce script produira soit notre message d'accueil, *soit* le message d'erreur. Mais il affichera toujours la ligne `Nombre d'arguments`.

```
$ ./new_script.sh
Veuillez ne saisir qu'un seul argument.
Nombre d'arguments : 0.
$ ./new_script.sh Carol
Hello Carol!
Nombre d'arguments : 1.
```

Remarquer la déclaration `if`. Nous avons utilisé `-eq` pour faire une *comparaison numérique*. Dans ce cas, nous vérifions que la valeur de `$#` est *égale* à un. Les autres comparaisons que nous pouvons effectuer sont :

**-ne**

Différent de

**-gt**

Supérieur à

**-ge**

Supérieur ou égal à

**-lt**

Inférieur à

**-le**

Inférieur ou égal à

## Exercices Guidés

1. L'utilisateur tape ce qui suit dans son shell :

```
$ PATH=~/.scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment variable.
ls: command not found
```

◦ Qu'a fait l'utilisateur ?

◦ Quelle commande permettra de combiner la valeur actuelle de PATH avec le nouveau répertoire `~/.scripts` ?

2. Considérez le script suivant. Remarquez qu'il utilise `elif` pour vérifier une deuxième condition :

```
> /!bin/bash

> fruit1 = Pommes
> fruit2 = Oranges

if [ $1 -lt $# ]
then
    echo "C'est comme comparer $fruit1 et $fruit2 "
> elif [$1 -gt $2 ]
then
>   echo '$fruit1 gagne !'
else
>   echo "Fruit2 gagne !"
> done
```

◦ Les lignes marquées d'un `>` contiennent des erreurs. Corrigez les erreurs.

3. Quel sera le résultat dans les situations suivantes ?



```
$ ./guided1.sh 3 0
```

```
$ ./guided1.sh 2 4
```

```
$ ./guided1.sh 0 1
```

## Exercices d'Exploration

1. Rédigez un script simple qui vérifiera si deux arguments exactement sont passés. Si c'est le cas, affichez les arguments dans l'ordre inverse. Prenons cet exemple (remarque : votre code peut être différent de celui-ci, mais il doit aboutir au même résultat) :

```
if [ $1 == $number ]
then
    echo "Vrai!"
fi
```

2. Ce code est correct, mais il ne s'agit pas d'une comparaison de chiffres. Recherchez sur Internet pour découvrir en quoi ce code est différent de l'utilisation de `-eq`.

3. Une variable d'environnement permet d'afficher le répertoire courant. Utilisez `env` pour découvrir le nom de cette variable.

4. En utilisant ce que vous avez appris dans les questions 2 et 3, écrivez un court script qui accepte un argument. Si un argument est passé, vérifiez si cet argument correspond au nom du répertoire courant. Si c'est le cas, affichez `oui`. Sinon, affichez `non`.

## Résumé

Dans cette section, vous avez appris :

- Comment créer et exécuter des scripts simples
- Comment utiliser un shebang pour spécifier un interprète
- Comment définir et utiliser les variables dans les scripts
- Comment gérer les arguments dans les scripts
- Comment construire des déclarations de `if`
- Comment comparer des nombres à l'aide d'opérateurs numériques

Commandes utilisées dans les exercices :

### **echo**

Affiche une chaîne de caractères sur la sortie standard.

### **env**

Affiche toutes les variables d'environnement sur la sortie standard.

### **which**

Affiche le chemin absolu d'une commande.

### **chmod**

Modifie les autorisations d'un fichier.

Variables spéciales utilisées dans les exercices :

### **\$1, \$2, ... \$9**

Contient des arguments positionnels transmis au script.

### **\$#**

Contient le nombre d'arguments passés au script.

### **\$PATH**

Contient les répertoires qui contiennent les exécutables utilisés par le système.

Opérateurs utilisés dans les exercices :

**-ne**

Différent de

**-gt**

Supérieur à

**-ge**

Supérieur ou égal à

**-lt**

Inférieur à

**-le**

Inférieur ou égal à

# Réponses aux Exercices Guidés

1. L'utilisateur tape dans son shell ce qui suit :

```
$ PATH=~/.scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment variable.
ls: command not found
```

- Qu'a fait l'utilisateur ?

L'utilisateur a écrasé le contenu de PATH avec le répertoire `~/scripts`. La commande `ls` ne peut plus être trouvée, car elle n'est pas contenue dans PATH. Notez que ce changement n'affecte que la session en cours, en se déconnectant et en se reconnectant les changements seront annulés.

- Quelle commande permettra de combiner la valeur actuelle de PATH avec le nouveau répertoire `~/scripts` ?

```
PATH=$PATH:~/scripts
```

2. Considérez le script suivant. Remarquez qu'il utilise `elif` pour vérifier une deuxième condition :

```
> /bin/bash

> fruit1 = Pommes
> fruit2 = Oranges

if [ $1 -lt $# ]
then
    echo "C'est comme comparer $fruit1 et $fruit2 !"
> elif [ $1 -gt $2 ]
then
>     echo '$fruit1 gagne !'
else
>     echo "Fruit2 gagne!"
> done
```

- Les lignes marquées d'un > contiennent des erreurs. Corrigez les erreurs.

```
#!/bin/bash

fruit1=Pommes
fruit2=Oranges

if [ $1 -lt $# ]
then
    echo "C'est comme comparer $fruit1 et $fruit2 !"
elif [ $1 -gt $2 ]
then
    echo "$fruit1 gagne !"
else
    echo "$fruit2 gagne !"
fi
```

### 3. Quel sera le résultat dans les situations suivantes ?

```
$ ./guided1.sh 3 0
```

Pommes gagne !

```
$ ./guided1.sh 2 4
```

Oranges gagne !

```
$ ./guided1.sh 0 1
```

C'est comme comparer Pommes et Oranges!

## Réponses aux Exercices d'Exploration

1. Rédigez un script simple qui vérifiera si deux arguments exactement sont passés. Si c'est le cas, affichez les arguments dans l'ordre inverse. Prenons cet exemple (remarque : votre code peut être différent de celui-ci, mais il doit aboutir au même résultat) :

```
if [ $1 == $number ]
then
    echo "True!"
fi
```

```
#!/bin/bash

if [ $# -ne 2 ]
then
    echo "Erreur"
else
    echo "$2 $1"
fi
```

2. Ce code est correct, mais il ne s'agit pas d'une comparaison de chiffres. Recherchez sur Internet pour découvrir en quoi ce code est différent de l'utilisation de `-eq`.

L'utilisation de `==` permettra de comparer les *chaînes de caractères*. C'est-à-dire que si les caractères des deux variables correspondent exactement, alors la condition est vraie.

<code>abc == abc</code>	<i>vraie</i>
<code>abc == ABC</code>	<i>faux</i>
<code>1 == 1</code>	<i>vraie</i>
<code>1+1 == 2</code>	<i>faux</i>

Les comparaisons de chaînes entraînent un comportement inattendu si vous faites des tests de chiffres.

3. Une variable d'environnement permet d'afficher le répertoire courant. Utilisez `env` pour découvrir le nom de cette variable.

PWD

4. En utilisant ce que vous avez appris dans les questions 2 et 3, écrivez un court script qui accepte un argument. Si un argument est passé, vérifiez si cet argument correspond au nom du répertoire courant. Si c'est le cas, affichez oui. Sinon, affichez non.

```
#!/bin/bash

if [ "$1" == "$PWD" ]
then
    echo "oui"
else
    echo "non"
fi
```





## 3.3 Leçon 2

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	3 La Puissance de la Ligne de Commande
<b>Objectif :</b>	3.3 Transformer les Commandes en Script
<b>Leçon :</b>	2 sur 2

### Introduction

Dans la dernière section, nous avons utilisé cet exemple simple pour démontrer l'écriture de script Bash :

```
#!/bin/bash

# Un script simple pour saluer un seul utilisateur.

if [ $# -eq 1 ]
then
    username=$1

    echo "Hello $username!"
else
    echo "Veuillez ne saisir qu'un seul argument."
fi
echo "Nombre d'arguments : $#."
```

- Tous les scripts doivent commencer par un *shebang*, qui définit le chemin vers l'interprète.
- Tous les scripts doivent comporter des commentaires pour décrire leur utilisation.
- Ce script particulier fonctionne avec un *argument*, qui est transmis au script lorsqu'il est appelé.
- Ce script contient une *déclaration if*, qui teste les conditions d'une variable intégrée `$#`. Cette variable contient le nombre d'arguments.
- Si le nombre d'arguments passés au script est égal à 1, alors la valeur du premier argument est passée à une nouvelle variable appelée `username` et le script envoie un message d'accueil à l'utilisateur. Sinon, un message d'erreur est affiché.
- Enfin, le script affiche le nombre d'arguments. Ceci est utile pour le débogage.

Ceci est un exemple utile pour commencer à expliquer certaines des autres caractéristiques des Scripts Bash.

## Codes de Sortie

Vous remarquerez que notre script présente deux résultats possibles : soit il affiche `"Hello <user>!"`, soit il affiche un message d'erreur. C'est tout à fait normal pour beaucoup de nos utilitaires de base. Prenons l'exemple de `cat`, que vous commencez sans doute à connaître.

Comparons une utilisation réussie de `cat` avec une situation où elle échoue. Nous vous rappelons que notre exemple ci-dessus est un script appelé `new_script.sh`.

```
$ cat -n new_script.sh

1  #!/bin/bash
2
3  # Un script simple pour saluer un seul utilisateur.
4
5  if [ $# -eq 1 ]
6  then
7      username=$1
8
9      echo "Hello $username!"
10 else
11     echo "Veuillez ne saisir qu'un seul argument."
12 fi
13 echo "Nombre d'arguments : $#."
```

Cette commande réussit, et vous remarquerez que l'option `-n` a également affiché des numéros de ligne. Ceux-ci sont très utiles pour déboguer les scripts, mais veuillez noter qu'ils *ne font pas* partie du script.

Nous allons maintenant vérifier la valeur d'une nouvelle variable intégrée `$?`. Pour l'instant, il suffit de noter la sortie :

```
$ echo $?  
0
```

Considérons maintenant une situation où `cat` échouera. Nous allons d'abord voir un message d'erreur, puis vérifier la valeur de `$?`.

```
$ cat -n dummyfile.sh  
cat: dummyfile.sh: No such file or directory  
$ echo $?  
1
```

L'explication de ce comportement est la suivante : toute exécution de l'utilitaire `cat` renvoie un *code de sortie*. Un code de sortie nous indiquera si la commande a réussi ou si elle a subi une erreur. Un code de sortie égal à *zéro* indique que la commande a réussi. Cela est vrai pour presque toutes les commandes Linux avec lesquelles vous travaillerez. Tout autre code de sortie indiquera une erreur quelconque. Le code de sortie de la *dernière commande exécutée* sera stocké dans la variable `$?`.

Les codes de sortie ne sont généralement pas vus par les utilisateurs humains, mais ils sont très utiles lors de l'écriture de scripts. Prenons l'exemple d'un script dans lequel nous pouvons copier des fichiers sur un lecteur réseau distant. La tâche de copie peut échouer de plusieurs façons : par exemple, notre machine locale peut ne pas être connectée au réseau, ou le disque distant peut être plein. En vérifiant le code de sortie de notre utilitaire de copie, nous pouvons signaler à l'utilisateur les problèmes rencontrés lors de l'exécution du script.

La mise en œuvre de codes de sortie est une très bonne pratique, c'est pourquoi nous allons le faire maintenant. Nous avons deux résultats possibles pour notre script, un succès ou un échec. Utilisons le zéro pour indiquer un succès et le un pour indiquer un échec.

```
1 #!/bin/bash  
2  
3 # Un script simple pour saluer un seul utilisateur.  
4
```

```

5  if [ $# -eq 1 ]
6  then
7      username=$1
8
9      echo "Hello $username!"
10     exit 0
11 else
12     echo "Veuillez ne saisir qu'un seul argument."
13     exit 1
14 fi
15 echo "Nombre d'arguments : $#."
```

```

$ ./new_script.sh Carol
Hello Carol!
$ echo $?
0
```

Remarquez que la commande `echo` sur la ligne 15 a été entièrement ignorée. L'utilisation de `exit` terminera le script immédiatement, donc cette ligne n'est jamais rencontrée.

## Gérer Plusieurs Arguments

Jusqu'à présent, notre script ne peut gérer qu'un seul nom d'utilisateur à la fois. Un nombre d'arguments supérieur à un entraînera une erreur. Voyons comment nous pouvons rendre ce script plus polyvalent.

Le premier réflexe d'un utilisateur pourrait être d'utiliser plus de variables positionnelles telles que `$2`, `$3`, etc. Malheureusement, nous ne pouvons pas anticiper le nombre d'arguments qu'un utilisateur pourrait choisir d'utiliser. Pour résoudre ce problème, il sera utile d'introduire davantage de variables intégrées.

Nous allons modifier la logique de notre script. Le fait d'avoir zéro argument devrait entraîner une erreur, mais tout autre nombre d'arguments devrait réussir. Ce nouveau script sera appelé `friendly2.sh`.

```

1  #!/bin/bash
2
3  # un script convivial pour saluer les utilisateurs
4
5  if [ $# -eq 0 ]
6  then
```

```

7     echo "Veuillez entrer au moins un utilisateur à saluer."
8     exit 1
9 else
10    echo "Hello $@"
11    exit 0
12 fi

```

```

$ ./friendly2.sh Carol Dave Henry
Hello Carol Dave Henry!

```

Il y a deux variables intégrées qui contiennent tous les arguments passés au script : `$@` et `$*`. La plupart du temps, les deux se comportent de la même manière. Bash va *analyser* les arguments et séparer chaque argument lorsqu'il rencontre un espace entre eux. En fait, le contenu de `$@` ressemble à ceci :

0	1	2
Carol	Dave	Henry

Si vous êtes familier avec d'autres langages de programmation, vous pouvez reconnaître ce type de variable comme un *tableau*. Les tableaux en Bash peuvent être créés simplement en mettant de l'espace entre des éléments comme la variable `FILES` dans le script `arraytest` ci-dessous :

```
FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"
```

Il contient une liste de nombreux éléments. Jusqu'à présent, cela n'est pas très utile, car nous n'avons pas encore introduit de moyen de traiter ces éléments individuellement.

## Les Boucles For

Référons-nous à l'exemple `arraytest` présenté précédemment. Si vous vous souvenez bien, dans cet exemple, nous spécifions un tableau appelé `FILES`. Ce dont nous avons besoin, c'est d'un moyen de "déballer" cette variable et d'accéder à chaque valeur individuelle, l'une après l'autre. Pour ce faire, nous utiliserons une structure appelée *boucle for*, qui est présente dans tous les langages de programmation. Nous nous référerons à deux variables : l'une est la plage et l'autre est la valeur individuelle sur laquelle nous travaillons actuellement. Ceci est le script dans son intégralité :

```
#!/bin/bash
```

```
FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"
```

```
for file in $FILES
do
    ls -lh $file
done
```

```
$ ./arraytest
```

```
lrwxrwxrwx 1 root root 10 Apr 24 11:02 /usr/sbin/accept -> cupsaccept
-rwxr-xr-x 1 root root 54K Mar 22 14:32 /usr/sbin/pwck
-rwxr-xr-x 1 root root 43K Jan 14 07:17 /usr/sbin/chroot
```

Si vous vous référez à nouveau à l'exemple `friendly2.sh` ci-dessus, vous pouvez voir que nous travaillons avec un ensemble de valeurs contenues dans une seule variable `$@`. Par souci de clarté, nous appellerons cette dernière variable `username`. Notre script ressemble maintenant à ceci :

```
1  #!/bin/bash
2
3  # un script convivial pour saluer les utilisateurs
4
5  if [ $# -eq 0 ]
6  then
7      echo "Veuillez entrer au moins un utilisateur à saluer."
8      exit 1
9  else
10     for username in $@
11     do
12         echo "Hello $username!"
13     done
14     exit 0
15 fi
```

Rappelez-vous que la variable que vous définissez ici peut être nommée comme vous le souhaitez, et que toutes les lignes à l'intérieur de `do... done` s'exécuteront une fois pour chaque élément du tableau. Observons la sortie de notre script :

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
```

Supposons maintenant que nous voulions donner à notre sortie un aspect un peu plus humain. Nous voulons que notre salutation soit sur une seule ligne.

```
1  #!/bin/bash
2
3  # un script convivial pour saluer les utilisateurs
4
5  if [ $# -eq 0 ]
6  then
7      echo "Veuillez entrer au moins un utilisateur à saluer."
8      exit 1
9  else
10     echo -n "Hello $1"
11     shift
12     for username in $@
13     do
14         echo -n ", et $username"
15     done
16     echo "!"
17     exit 0
18 fi
```

Quelques notes :

- L'utilisation de `-n` avec `echo` *supprimera le saut de ligne* après l'affichage. Cela signifie que tous les echos seront affichés sur la même ligne, et que le saut de ligne ne sera affiché qu'après le `!` de la ligne 16.
- La commande `shift` supprimera le premier élément de notre tableau, afin que ceci :

0	1	2
Carol	Dave	Henry

Devient ceci :

0	1
Dave	Henry

Observons le résultat :

```
$ ./friendly2.sh Carol
```

```
Hello Carol!
$ ./friendly2.sh Carol Dave Henry
Hello Carol, et Dave, et Henry!
```

## Usage d'Expressions Régulières pour Vérifier des Erreurs

Il est possible que nous voulions vérifier tous les arguments que l'utilisateur saisit. Par exemple, nous voulons peut-être nous assurer que tous les noms transmis à `friendly2.sh` *ne* contiennent *que des lettres*, et que tout caractère spécial ou chiffre provoquera une erreur. Pour effectuer cette vérification d'erreur, nous utiliserons `grep`.

Rappelons que nous pouvons utiliser des expressions régulières avec `grep`.

```
$ echo Animal | grep "^[A-Za-z]*$"
Animal
$ echo $?
0
```

```
$ echo 4n1m1 | grep "^[A-Za-z]*$"
$ echo $?
1
```

Le `^` et le `$` indiquent respectivement le début et la fin de la ligne. Le `[A-Za-z]` indique une série de lettres, en majuscules ou en minuscules. Le `*` est un *quantificateur* et modifie notre plage de lettres de manière que nous fassions correspondre de zéro à plusieurs lettres. En résumé, notre `grep` réussira si la saisie *ne* comporte *que* des lettres, et échouera dans le cas contraire.

Ensuite, il faut noter que `grep` renvoie des codes de sortie en fonction de l'existence ou non d'une correspondance. Une correspondance positive renvoie `0`, et une absence de correspondance renvoie un `1`. Nous pouvons utiliser cela pour tester nos arguments à l'intérieur de notre script.

```
1 #!/bin/bash
2
3 # un script convivial pour saluer les utilisateurs
4
5 if [ $# -eq 0 ]
6 then
7     echo "Veuillez entrer au moins un utilisateur à saluer."
8     exit 1
9 else
```



```
10  for username in $@
11  do
12      echo $username | grep "^[A-Za-z]*$" > /dev/null
13      if [ $? -eq 1 ]
14      then
15          echo "ERREUR : Les noms ne doivent contenir que des lettres."
16          exit 2
17      else
18          echo "Hello $username!"
19      fi
20  done
21  exit 0
22  fi
```

Sur la ligne 12, nous redirigeons la sortie standard vers `/dev/null`, ce qui est une façon simple de la supprimer. Nous ne voulons pas voir de sortie de la commande `grep`, nous voulons seulement tester son code de sortie, ce qui se passe sur la ligne 13. Notez également que nous utilisons un code de sortie de 2 pour indiquer un argument non valide. Il est généralement bon d'utiliser différents codes de sortie pour indiquer différentes erreurs ; de cette façon, un utilisateur averti peut utiliser ces codes de sortie pour déboguer.

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
$ ./friendly2.sh 42 Carol Dave Henry
ERREUR : Les noms ne doivent contenir que des lettres.
$ echo $?
2
```

## Exercices Guidés

1. Lisez le contenu de `script1.sh` ci-dessous :

```
#!/bin/bash

if [ $# -lt 1 ]
then
    echo "Ce script nécessite au moins 1 argument."
    exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
    echo "pas de gateau pour toi !"
    exit 2
fi

echo "voici ton gateau !"
exit 0
```

Quel est le résultat de ces commandes ?

◦ `./script1.sh`

◦ `echo $?`

◦ `./script1.sh cake`

◦ `echo $?`

◦ `./script1.sh CAKE`

◦ `echo $?`

2. Lisez le contenu du fichier `script2.sh` :

```
for filename in $1/*.txt
do
    cp $filename $filename.bak
done
```

Décrivez le but de ce script tel que vous le comprenez.

---

## Exercices d'Exploration

1. Créez un script qui prendra un nombre quelconque d'arguments de l'utilisateur et n'affichera que les arguments qui sont des nombres supérieurs à 10.

## Résumé

Dans cette section, vous avez appris :

- Quels sont les codes de sortie, leur signification et comment les mettre en œuvre
- Comment vérifier le code de sortie d'une commande
- Ce que sont les boucles `for` et comment les utiliser avec des tableaux
- Comment utiliser `grep`, les expressions régulières et les codes de sortie pour vérifier les entrées de l'utilisateur dans les scripts.

Commandes utilisées dans les exercices :

### **shift**

Ceci supprimera le premier élément d'un tableau.

Variables spéciales :

### **\$?**

Contient le code de sortie de la dernière commande exécutée.

### **\$@, \$\***

Contient tous les arguments passés au script, sous forme de tableau.

# Réponses aux Exercices Guidés

1. Lisez le contenu de `script1.sh` ci-dessous :

```
#!/bin/bash

if [ $# -lt 1 ]
then
    echo "Ce script nécessite au moins 1 argument."
    exit 1
fi

echo $1 | grep "[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
    echo "pas de gateau pour toi !"
    exit 2
fi

echo "voici ton gateau !"
exit 0
```

Quel est le résultat de ces commandes ?

- Commande : `./script1.sh`

Sortie : `Ce script nécessite au moins 1 argument.`

- Commande : `echo $?`

Sortie : `1`

- Commande : `./script1.sh gateau`

Sortie : `pas de gateau pour toi !`

- Commande : `echo $?`

Sortie : `2`

- Commande : `./script1.sh GATEAU`

Sortie : `voici ton gateau !`

◦ Commande : `echo $?`

Sortie : `0`

2. Lisez le contenu du fichier `script2.sh` :

```
for filename in $1/*.txt
do
    cp $filename $filename.bak
done
```

Décrivez le but de ce script tel que vous le comprenez.

Ce script fera des copies de sauvegarde de tous les fichiers se terminant par `.txt` dans un sous-répertoire défini dans le premier argument.

# Réponses aux Exercices d'Exploration

1. Créez un script qui prendra un nombre quelconque d'arguments de l'utilisateur et n'affichera que les arguments qui sont des nombres supérieurs à 10.

```
#!/bin/bash

for i in $@
do
    echo $i | grep "^[0-9]*$" > /dev/null
    if [ $? -eq 0 ]
    then
        if [ $i -gt 10 ]
        then
            echo -n "$i "
        fi
    fi
done
echo ""
```





## **Sujet 4 : Le système d'exploitation Linux**



**Linux  
Professional  
Institute**

## **4.1 Choix d'un système d'exploitation**

### **Référence aux objectifs de LPI**

Linux Essentials version 1.6, Exam 010, Objective 4.1

### **Valeur**

1

### **Domaines de connaissance les plus importants**

- Différences entre Windows, OS X et Linux
- Gestion du cycle de vie des distributions

### **Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif**

- Interface graphique par rapport à la ligne de commande, configuration d'un poste de travail
- Cycles de maintenance, versions bêta et stables



## 4.1 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version:</b>	1.6
<b>Thème :</b>	4 Le Système d'Exploitation Linux
<b>Objectif :</b>	4.1 Choisir un Système d'Exploitation
<b>Leçon:</b>	1 sur 1

### Introduction

Peu importe que vous utilisiez votre système informatique personnel, à l'université ou dans une entreprise, une décision doit encore être prise quant au système d'exploitation que vous utiliserez. Cette décision peut être prise par vous, surtout s'il s'agit de votre ordinateur, mais vous pouvez également être responsable du choix des systèmes de votre entreprise. Comme toujours, le fait d'être bien informé des choix possibles vous aidera à prendre une décision responsable. Dans cette leçon, nous voulons vous aider à vous tenir informé des choix de systèmes d'exploitation que vous pourriez envisager.

### Qu'est-ce qu'un Système d'Exploitation

L'une des premières choses dont nous devons être sûrs avant d'entamer notre voyage dans le choix d'un système d'exploitation est de comprendre ce que nous entendons par ce terme. Le système d'exploitation se trouve au cœur de votre ordinateur et permet aux applications de fonctionner à l'intérieur et au-dessus de celui-ci. En outre, le système d'exploitation contient les pilotes permettant d'accéder au matériel de l'ordinateur, comme les disques et les partitions, les écrans, les claviers, les cartes réseau, etc. Nous abrégons souvent le terme système d'exploitation par OS (Operating System). Aujourd'hui, il existe de nombreux systèmes d'exploitation, tant pour

une utilisation professionnelle de l'ordinateur que pour un usage personnel. Si nous voulons simplifier la sélection qui nous est offerte, nous pouvons regrouper les sélections comme suit :

- Systèmes d'exploitation basés sur Linux
  - Linux entreprise
  - Linux grand public
- UNIX
- macOS
- Systèmes d'exploitation basés sur Windows
  - Serveurs Windows
  - Bureaux Windows

## Choisir une Distribution Linux

### Le Noyau Linux et les Distributions Linux

Lorsque l'on parle de distributions Linux, le système d'exploitation est Linux. Linux est le *noyau* et le cœur de chaque distribution Linux. Le logiciel du noyau Linux est maintenu par un groupe d'individus, dirigé par Linus Torvalds. Torvalds est employé par un consortium industriel appelé [The Linux Foundation](#) pour travailler sur le noyau Linux.

#### NOTE

Le noyau Linux a été développé pour la première fois par Linus Torvalds, un étudiant finlandais, en 1991. En 1992, la première version du noyau sous licence GNU General Public License version 2 (GPLv2) était la version 0.12.

### Noyau Linux

Comme nous l'avons mentionné, toutes les distributions Linux utilisent le même système d'exploitation, Linux.

### Distribution Linux

Lorsque les gens parlent de Red Hat Linux, ou Ubuntu Linux, ils font référence à la *distribution Linux*. La distribution Linux sera livrée avec un noyau Linux et un environnement qui rendra le noyau utile de manière que nous puissions interagir avec lui. Au minimum, nous aurons besoin d'un shell en ligne de commande tel que Bash et d'un ensemble de commandes de base nous permettant d'accéder au système et de le gérer. Souvent, bien sûr, la distribution Linux disposera d'un environnement de bureau complet comme Gnome ou KDE.

Même si chaque distribution Linux utilise le système d'exploitation Linux, les distributions

peuvent varier et varient effectivement en fonction de la version du système d'exploitation utilisée. Par-là, nous entendons la version du *noyau Linux qui est utilisée* au démarrage de la distribution.

Si vous avez accès à une ligne de commande Linux en ce moment, vous pouvez facilement vérifier la version du noyau Linux que vous utilisez en lisant la *version du noyau* :

```
$ uname -r  
4.15.0-1019-aws
```

## Types de Distributions Linux

Le choix de toujours utiliser la dernière version du noyau Linux peut sembler évident, mais ce n'est pas aussi simple que cela. Nous pouvons vaguement classer les distributions Linux en trois catégories :

- Distributions Linux de Grade Enterprise
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Distributions Linux de Grade Grand Public
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Distributions Linux Expérimentales et Hackers
  - Arch
  - Gentoo

Il ne s'agit bien sûr que d'un très petit sous-ensemble de distributions possibles, mais l'importance réside dans la différence entre les distributions *entreprises*, *grand public* et *expérimentales* et dans la raison d'être de chacune.

## Linux de Grade Enterprise

Les distributions telles que CentOS (*Community Enterprise OS*) sont conçues pour être déployées dans les grandes organisations utilisant du matériel d'entreprise. Les besoins des grandes entreprises sont très différents des besoins des petites entreprises, des amateurs ou des utilisateurs à domicile. Afin de garantir la disponibilité de leurs services, les utilisateurs en entreprises ont des exigences plus élevées en ce qui concerne la stabilité de leur matériel et de leurs logiciels. C'est pourquoi les distributions Linux d'entreprise ont tendance à inclure des versions plus anciennes du noyau et d'autres logiciels, dont on sait qu'ils fonctionnent de manière fiable. Souvent, les distributions portent des mises à jour importantes, comme des correctifs de sécurité, vers ces versions stables. En retour, les distributions Linux d'entreprise peuvent ne pas prendre en charge le matériel grand public le plus récent et fournir des versions plus anciennes de logiciels. Cependant, comme pour les distributions Linux grand public, les entreprises ont tendance à choisir des composants matériels matures et à développer leurs services sur des versions logicielles stables.

## Linux pour le Grand Public

Les distributions telles qu'Ubuntu sont plus ciblées par les petites entreprises ou les particuliers et les utilisateurs amateurs. En tant que telles, elles sont également susceptibles d'utiliser le matériel le plus récent que l'on trouve sur les systèmes grand public. Ces systèmes auront besoin des derniers pilotes pour tirer le meilleur parti du nouveau matériel, mais il est peu probable que la maturité du matériel et des pilotes réponde aux besoins des grandes entreprises. Pour le marché grand public, cependant, le dernier noyau est exactement ce dont on a besoin avec les pilotes les plus modernes, même s'ils sont peu testés. Les noyaux Linux les plus récents disposeront des derniers pilotes pour prendre en charge le matériel le plus récent qui sera probablement utilisé. Avec l'évolution que nous constatons avec Linux sur le marché des jeux, il est extrêmement important que les derniers pilotes soient disponibles pour ces utilisateurs.

### NOTE

Certaines distributions comme Ubuntu fournissent à la fois des versions grand public qui contiennent des logiciels récents et reçoivent des mises à jour pendant une période assez courte, ainsi que des versions dites de support à long terme, LTS en abrégé, qui sont plus adaptées aux environnements d'entreprise.

## Distributions Linux Expérimentales et hackers

Des distributions telles que Arch Linux ou Gentoo Linux sont à la pointe de la technologie. Elles contiennent les versions les plus récentes des logiciels, même si ces versions contiennent encore des bugs et des fonctionnalités non testées. En retour, ces distributions ont tendance à utiliser un modèle de publication à roulement qui leur permet de fournir des mises à jour à tout moment. Ces distributions sont utilisées par des utilisateurs avancés qui veulent toujours recevoir les logiciels les plus récents et qui sont conscients que les fonctionnalités peuvent être

interrompues à tout moment et sont capables de réparer leurs systèmes dans de tels cas.

En bref, si vous considérez Linux comme votre système d'exploitation, si vous utilisez du matériel de qualité professionnelle sur vos serveurs ou vos ordinateurs de bureau, vous pouvez utiliser des distributions Linux de qualité professionnelle ou de qualité grand public. Si vous utilisez du matériel grand public et que vous devez tirer le meilleur parti des dernières innovations matérielles, il est probable que vous aurez besoin d'une distribution Linux similaire pour répondre aux besoins du matériel.

Certaines distributions de Linux sont liées entre elles. Ubuntu, par exemple, est basée sur Debian Linux et utilise le même système de gestion des paquets, DPKG. Fedora, autre exemple, est un banc d'essai pour RedHat Enterprise Linux, où les caractéristiques potentielles des futures versions de RHEL peuvent être explorées avant leur disponibilité dans la distribution d'entreprise.

En plus des distributions que nous avons mentionnées ici, il existe de nombreuses autres distributions Linux. Un des avantages de Linux en tant que logiciel open source est que de nombreuses personnes peuvent développer ce à quoi elles pensent que Linux devrait ressembler. Nous avons donc plusieurs centaines de distributions. Pour voir d'autres distributions Linux, vous pouvez choisir de visiter le [site web The Distro Watch](#), les responsables du site listent les 100 meilleurs téléchargements de distributions Linux, ce qui vous permet de comparer et de voir ce qui est actuellement populaire.

## Cycle de Vie du Support Linux

Comme vous pouvez vous en douter, les distributions Linux d'entreprise ont une durée de vie plus longue que les éditions grand public ou communautaires de Linux. Par exemple, Red Hat Enterprise Linux a une durée de support de 10 ans. Red Hat Enterprise Linux 8 a été lancé en mai 2019, tandis que les mises à jour logicielles et le support sont disponibles jusqu'en mai 2029.

Les éditions grand public ne bénéficient souvent que du soutien de la communauté par le biais de forums. Les mises à jour logicielles sont souvent disponibles pour 3 versions. Si nous prenons l'exemple d'Ubuntu, au moment où nous écrivons, la version 19.04 est la dernière disponible ayant des mises à jour jusqu'à la sortie de la version 19.10 et s'arrêtant en janvier 2020. Ubuntu fournit également des éditions avec un support à long terme, connues sous le nom d'éditions *LTS*, qui ont 5 ans de support à partir de la version originale. La version LTS actuelle est la 18.04, qui comprendra des mises à jour logicielles jusqu'en 2023. Ces versions LTS font d'Ubuntu une option possible pour l'entreprise avec un support commercial disponible auprès de Canonical (la société derrière la marque Ubuntu) ou des sociétés de conseil indépendantes.

### NOTE

Les distributions Ubuntu utilisent des numéros de version basés sur la date au format YY.MM : par exemple, la version 19.04 est sortie en avril 2019.

## Linux comme Système de Bureau

Utiliser Linux comme système de bureau peut être plus difficile dans une entreprise où le support bureautique se concentre sur les offres de systèmes d'exploitation commerciaux. Cependant, ce n'est pas seulement le support qui peut s'avérer difficile. Une entreprise cliente peut également avoir fait d'importants investissements dans des solutions logicielles qui les relient à des systèmes d'exploitation spécifiques pour ordinateurs de bureau. Cela dit, il existe de nombreux exemples d'intégration de postes de travail Linux dans de grandes organisations, des entreprises comme Amazon ayant même leur propre distribution Linux [Amazon Linux 2](#), utilisée sur leur plate-forme AWS cloud, mais aussi en interne pour les serveurs et les postes de travail.

Utiliser Linux dans une petite entreprise ou personnellement devient beaucoup plus facile et peut être une expérience enrichissante, en supprimant le besoin de licences et en vous ouvrant les yeux sur la richesse des logiciels libres et open source disponibles pour Linux. Vous constaterez également qu'il existe de nombreux environnements de bureau différents. Les plus courants sont Gnome et KDE, mais il en existe beaucoup d'autres. La décision dépend de vos préférences personnelles.

## Utilisation de Linux sur les Serveurs

L'utilisation de Linux comme système d'exploitation de votre serveur est une pratique courante dans le secteur des entreprises. Les serveurs sont entretenus par des ingénieurs spécialisés dans Linux. Ainsi, même avec des milliers d'utilisateurs, les utilisateurs peuvent rester ignorants des serveurs auxquels ils se connectent. Le système d'exploitation des serveurs n'est pas important pour eux et, en général, les applications clientes ne diffèrent pas entre Linux et les autres systèmes d'exploitation en arrière-plan. Il est également vrai que plus les applications sont virtualisées ou conteneurisées dans des clouds locaux et distants, plus le système d'exploitation est masqué et plus le système d'exploitation intégré est susceptible d'être Linux.

## Linux dans le Cloud

Une autre possibilité de se familiariser avec Linux est de déployer Linux dans l'un des nombreux clouds publics disponibles. La création d'un compte auprès de l'un des nombreux autres fournisseurs clouds vous permettra de déployer rapidement et facilement de nombreuses distributions Linux différentes.

## Systèmes d'Exploitation Non Linux

Oui, aussi incroyable que cela puisse paraître, il existe des systèmes d'exploitation qui ne sont pas basés sur le noyau Linux. Bien sûr, au fil des ans, il y en a eu beaucoup et certains ont été abandonnés, mais il y a encore d'autres choix qui s'offrent à vous. Que ce soit personnellement ou



au bureau.

## Unix

Avant que nous ayons Linux comme système d'exploitation, il y avait Unix. Unix était vendu avec le matériel et aujourd'hui encore, plusieurs Unix commerciaux tels que AIX et HP-UX sont disponibles sur le marché. Alors que Linux s'est fortement inspiré d'Unix (et du manque de disponibilité de ce dernier pour certains matériels), la famille des systèmes d'exploitation BSD est directement basée sur Unix. Aujourd'hui, FreeBSD, NetBSD et OpenBSD, ainsi que certains autres systèmes BSD connexes, sont disponibles en tant que logiciels libres.

Unix était très utilisé en entreprise, mais nous avons constaté un déclin de la fortune d'Unix avec la croissance de Linux. Avec la croissance de Linux et la croissance des offres de support aux entreprises, nous avons vu Unix commencer lentement à disparaître. Solaris, originaire de Sun avant de passer à Oracle, a récemment disparu. C'était l'un des plus grands systèmes d'exploitation Unix utilisés par les entreprises de télécommunications, annoncé comme étant un *Unix de qualité Telco*.

Les systèmes d'exploitation Unix comprennent :

- AIX
- FreeBSD, NetBSD, OpenBSD
- HP-UX
- Irix
- Solaris

## macOS

macOS (anciennement OS X) d'Apple remonte à 2001. Basé en grande partie sur BSD Unix, et utilisant le shell en ligne de commande Bash, c'est un système convivial à utiliser si vous êtes habitué à utiliser les systèmes d'exploitation Unix ou Linux. Si vous utilisez macOS, vous pouvez ouvrir l'application du terminal pour accéder à la ligne de commande. En exécutant à nouveau la même commande `uname`, nous pouvons vérifier le système d'exploitation installé :

```
$ uname -s
Darwin
```

### NOTE

Nous utilisons l'option `-s` dans ce cas pour retourner le nom du système d'exploitation. Nous utilisions auparavant `-r` pour renvoyer le numéro de version

du noyau.

## Microsoft Windows

Nous pouvons toujours dire que la majorité des ordinateurs de bureau et portables seront basés sur Windows. Ce système d'exploitation a connu un véritable succès et a dominé le marché des ordinateurs de bureau pendant des années. Bien qu'il s'agisse d'un logiciel propriétaire et qu'il ne soit pas gratuit, la licence du système d'exploitation est souvent incluse lorsque vous achetez le matériel, ce qui en fait un choix facile à faire. Bien entendu, Windows est largement pris en charge par les fournisseurs de matériel et de logiciels, et de nombreuses applications open source sont également disponibles pour Windows. L'avenir de Windows ne semble pas aussi brillant qu'il l'a été. Les ordinateurs de bureau et les ordinateurs portables étant de moins en moins vendus, l'accent est mis sur le marché des tablettes et des téléphones. Ce marché a été dominé par Apple et Android et il est difficile pour Microsoft d'y gagner du terrain.

En tant que plate-forme serveur, Microsoft permet désormais à ses clients de choisir entre une version GUI (*Graphical User Interface*) et une version en ligne de commande uniquement. La séparation de l'interface graphique et de la ligne de commande est importante. La plupart du temps, l'interface graphique des anciens serveurs Microsoft sera chargée, mais personne ne l'utilisera. Imaginez un contrôleur de domaine Active Directory... les utilisateurs l'utilisent tout le temps pour s'authentifier sur le domaine, mais il est géré à distance depuis les bureaux des administrateurs et non depuis le serveur.

## Exercices Guidés

1. Quel projet constitue le composant commun à toutes les distributions Linux ?

CentOS	
Red Hat	
Ubuntu	
Noyau Linux	
CoreOS	

2. Quel est le système d'exploitation utilisé par Apple pour MacOS ?

OS X	
OSX	
Darwin	
MacOS	

3. En quoi une distribution Linux diffère-t-elle du noyau Linux ?

Le noyau fait partie d'une distribution, la distribution a un ensemble d'applications qui entourent le noyau afin de le rendre utile	
Le noyau est la distribution Linux	
Toutes les distributions qui utilisent le même noyau sont identiques	

4. Parmi les éléments suivants, lequel est un environnement de bureau sous Linux ?

Mint	
Elementary	
Zorin	
Gnome	

5. Quel composant d'un système d'exploitation permet l'accès au matériel ?

Pilotes	
Shells	
Service	
Application	

## Exercices d'Exploration

1. Retrouvez la sortie actuelle du noyau de votre système Linux si vous avez accès à la ligne de commande.

2. À l'aide de votre moteur de recherche préféré, localisez et identifiez les fournisseurs de services cloud publics qui sont à votre disposition. Il peut s'agir de AWS, Google Cloud, Rackspace et bien d'autres encore. Choisissez-en un et voyez quels systèmes d'exploitation y sont disponibles pour les déploiements.

## Résumé

Dans cette section, vous avez appris à faire la différence entre les différents systèmes d'exploitation couramment disponibles. Nous avons discuté :

- des systèmes d'exploitation basés sur Linux
- d'UNIX
- de macOS
- des systèmes d'exploitation basés sur Windows

Dans la catégorie Linux, nous pourrions décomposer la sélection en distributions avec un support à long terme et celles avec un cycle de support plus court. Les versions LTS étant plus adaptées à l'entreprise et le support à court terme étant destiné aux utilisateurs privés et aux amateurs.

- Distributions Linux de Grade Enterprise
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Distributions Linux de Grade Grand Public
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Distributions Linux Expérimentales et Hackers
  - Arch
  - Gentoo

# Réponses aux Exercices Guidés

1. Quel projet constitue le composant commun à toutes les distributions Linux ?

CentOS	
Red Hat	
Ubuntu	
Noyau Linux	X
CoreOS	

2. Quel est le système d'exploitation utilisé par Apple pour OS X ?

OS X	
OSX	
Darwin	X
MacOS	

3. En quoi une distribution Linux diffère-t-elle du noyau Linux ?

Le noyau fait partie d'une distribution, la distribution a un ensemble d'applications qui entourent le noyau afin de le rendre utile	X
Le noyau est la distribution Linux	
Toutes les distributions qui utilisent le même noyau sont identiques	

4. Parmi les éléments suivants, lequel est un environnement de bureau sous Linux ?

Mint	
Elementary	
Zorin	
Gnome	X

5. Quel composant d'un système d'exploitation permet l'accès au matériel ?

Pilotes	X
Shells	
Service	
Application	



## Réponses aux Exercices d'Exploration

1. Retrouvez la sortie actuelle du noyau de votre système Linux si vous avez accès à la ligne de commande.

```
$ uname -r  
4.15.0-47-generic
```

2. À l'aide de votre moteur de recherche préféré, localisez et identifiez les fournisseurs de services cloud publics qui sont à votre disposition. Il peut s'agir de AWS, Google Cloud, Rackspace et bien d'autres encore. Choisissez-en un et voyez quels systèmes d'exploitation y sont disponibles pour les déploiements.

AWS, par exemple, vous permet de déployer de nombreuses distributions Linux telles que Debian, Red Hat, SUSE ou Ubuntu ainsi que Windows.



## 4.2 Compréhension du matériel informatique

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 4.2

### Valeur

2

### Domaines de connaissance les plus importants

- Matériel informatique

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- Cartes mères, processeurs, alimentations, disques optiques, périphériques
- Disques durs, SSD et partitions, `/dev/sd*``
- Pilotes de périphériques



## 4.2 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	4 Le Système d'Exploitation Linux
<b>Objectif :</b>	4.2 Comprendre le Matériel Informatique
<b>Leçon:</b>	1 sur 1

### Introduction

Sans matériel, le logiciel n'est rien d'autre qu'une autre forme de littérature. Le matériel traite les commandes décrites par le logiciel et fournit des mécanismes de stockage, d'entrée et de sortie. Même le cloud est en fin de compte soutenu par le matériel.

En tant que système d'exploitation, l'une des responsabilités de Linux est de fournir des logiciels avec des interfaces pour accéder au matériel d'un système. La plupart des spécificités de configuration dépassent le cadre de cette leçon. Cependant, les utilisateurs sont souvent préoccupés par les performances, la capacité et d'autres facteurs du matériel du système, car ils ont une incidence sur la capacité d'un système à prendre en charge de manière adéquate des applications spécifiques. Cette leçon aborde le matériel en tant qu'éléments physiques distincts utilisant des connecteurs et des interfaces standard. Les normes sont relativement statiques. Mais le facteur de forme, les performances et les caractéristiques de capacité du matériel sont en constante évolution. Indépendamment de la manière dont les changements peuvent brouiller les distinctions physiques, les aspects conceptuels du matériel décrits dans cette leçon restent d'actualité.

#### NOTE

À différents moments de cette leçon, des exemples de lignes de commande sont

utilisés pour montrer comment accéder aux informations sur le matériel. La plupart des exemples sont tirés d'un Raspberry Pi B+ mais devraient s'appliquer à la plupart des systèmes. La compréhension de ces commandes n'est pas nécessaire pour comprendre ce matériel.

## Les Alimentations Électriques

Tous les composants actifs d'un système informatique ont besoin d'électricité pour fonctionner. Malheureusement, la plupart des sources d'électricité ne sont pas appropriées. Le matériel des systèmes informatiques nécessite des tensions spécifiques avec des tolérances relativement étroites. Ce qui n'est pas le cas de votre prise murale locale.

Les alimentations électriques normalisent les sources d'énergie disponibles. Les exigences de tension normalisées permettent aux fabricants de créer des composants matériels qui peuvent être utilisés dans des systèmes partout dans le monde. Les blocs d'alimentation de bureau ont tendance à utiliser l'électricité des prises murales comme source. Les blocs d'alimentation des serveurs sont généralement plus critiques et peuvent souvent être connectés à plusieurs sources pour garantir leur fonctionnement en cas de défaillance d'une source.

La consommation d'énergie génère de la chaleur. Une chaleur excessive peut entraîner un fonctionnement lent ou même une défaillance des composants du système. La plupart des systèmes sont équipés d'une forme de ventilateur pour faire circuler l'air afin d'obtenir un refroidissement plus efficace. Les composants tels que les processeurs génèrent souvent de la chaleur que le flux d'air seul ne peut pas dissiper. Ces composants chauds fixent des ailettes spéciales appelées dissipateurs de chaleur pour aider à dissiper la chaleur qu'ils génèrent. Les dissipateurs de chaleur ont souvent leur propre petit ventilateur pour assurer un flux d'air adéquat.

## La Carte Mère

Tout le matériel d'un système doit être interconnecté. Une carte mère normalise cette interconnexion en utilisant des connecteurs et des facteurs de forme standardisés. Elle fournit également un support pour la configuration et les besoins électriques de ces connecteurs.

Il existe un grand nombre de configurations de cartes mères. Elles prennent en charge différents processeurs et systèmes de mémoire. Elles ont différentes combinaisons de connecteurs standardisés. Et elles s'adaptent aux différentes tailles des boîtiers qui les contiennent. Hormis peut-être la possibilité de connecter des dispositifs externes spécifiques, la configuration de la carte mère est effectivement transparente pour les utilisateurs. Les administrateurs sont surtout exposés à la configuration de la carte mère lorsqu'il est nécessaire d'identifier des dispositifs

spécifiques.

Lors de la première mise sous tension, un matériel spécifique à la carte mère doit être configuré et initialisé avant que le système puisse fonctionner. Les cartes mères utilisent une programmation stockée dans une mémoire non volatile appelée firmware (*microprogramme*) pour traiter le matériel spécifique à la carte mère. La forme originale du firmware de la carte mère était connue sous le nom de BIOS (*Basic Input/Output System*). Au-delà des paramètres de configuration de base, le BIOS était principalement responsable de l'identification, du chargement et du transfert des opérations vers un système d'exploitation tel que Linux. Avec l'évolution du matériel, les firmwares se sont développés pour prendre en charge des disques plus grands, des diagnostics, des interfaces graphiques, la mise en réseau et d'autres capacités avancées indépendantes de tout système d'exploitation chargé. Les premières tentatives pour faire progresser les firmwares au-delà du BIOS de base étaient souvent spécifiques à un fabricant de cartes mères. Intel a défini une norme pour les firmwares avancés, connue sous le nom d'EFI (*Extensible Firmware Interface*). Intel a contribué à la création de l'EFI au sein d'un organisme de normalisation pour créer l'UEFI (*Unified Extensible Firmware Interface*). Aujourd'hui, la plupart des cartes mères utilisent l'UEFI. Le BIOS et l'EFI ne sont presque jamais disponibles sur les systèmes récents. Quoiqu'il en soit, la plupart des gens appellent encore le firmware de la carte mère BIOS.

Il existe très peu de paramètres de firmwares intéressant les utilisateurs généraux, de sorte que seules les personnes responsables de la configuration matérielle du système doivent généralement s'occuper des firmwares et de leurs paramètres. L'une des rares options couramment modifiées est l'activation des extensions de virtualisation des processeurs modernes.

## La Mémoire

La mémoire du système contient les données et le code de programme des applications en cours d'exécution. Lorsqu'ils parlent de la mémoire de l'ordinateur, la plupart des gens font référence à cette mémoire système. Un autre terme courant utilisé pour la mémoire système est l'acronyme RAM (*Random Access Memory*) ou une variante de cet acronyme. Parfois, des références au conditionnement physique de la mémoire système comme DIMM, SIMM ou DDR sont également utilisées.

Physiquement, la mémoire système est généralement conditionnée sur des modules de circuits imprimés individuels qui se branchent sur la carte mère. La taille des modules de mémoire individuels varie actuellement entre 2 Go et 64 Go. Pour la plupart des applications générales, 4 Go est le minimum de mémoire système que les gens devraient envisager. Pour les postes de travail individuels, 16 Go sont généralement plus que suffisants. Cependant, même 16 Go peuvent être une limite pour les utilisateurs qui utilisent des jeux, des vidéos ou des applications audios haut de gamme. Les serveurs ont souvent besoin de 128 Go, voire 256 Go de mémoire pour supporter efficacement les charges des utilisateurs.

Pour l'essentiel, Linux permet aux utilisateurs de traiter la mémoire système comme une boîte noire. Quand une application est lancée, Linux se charge d'allouer la mémoire système nécessaire. Linux libère la mémoire pour qu'elle puisse être utilisée par d'autres applications lorsqu'une application se termine. Mais que se passe-t-il si une application nécessite plus que la mémoire système disponible ? Dans ce cas, Linux déplace les applications inutilisées de la mémoire système vers une zone de disque spéciale appelée espace d'échange (espace swap). Linux déplace les applications inutilisées de l'espace d'échange du disque vers la mémoire système lorsqu'elles doivent s'exécuter.

Les systèmes sans matériel vidéo dédié utilisent souvent une partie de la mémoire système (souvent 1 Go) pour servir de stockage d'affichage vidéo. Cela réduit la mémoire système effective. Le matériel vidéo dédié possède généralement sa propre mémoire séparée qui n'est pas disponible en tant que mémoire système.

Il existe plusieurs façons d'obtenir des informations sur la mémoire du système. En tant qu'utilisateur, la quantité totale de mémoire disponible et utilisée est généralement la valeur d'intérêt. Une source d'information serait d'exécuter la commande `free` avec le paramètre `-m` pour convertir la sortie en mégaoctets :

```
$ free -m
```

	total	used	free	shared	buff/cache	available
Mem:	748	37	51	14	660	645
Swap:	99	0	99			

La première ligne précise la mémoire totale disponible pour le système (`total`), la mémoire utilisée (`used`) et la mémoire libre (`free`). La deuxième ligne affiche ces informations pour l'espace d'échange (`swap`). La mémoire indiquée comme `shared` et `buff/cache` est actuellement utilisée pour d'autres fonctions du système, bien que la quantité indiquée dans `available` puisse être utilisée pour l'application.

## Les Processeurs

Le mot "processeur" implique que quelque chose est en cours de traitement. Dans les ordinateurs, la majeure partie de ce traitement porte sur des signaux électriques. Généralement, ces signaux sont traités comme ayant une des valeurs binaires 1 ou 0.

Lorsque les gens parlent d'ordinateurs, ils utilisent souvent le terme "word processor" de façon interchangeable avec l'acronyme "CPU" (*Central Processing Unit*). Ce qui n'est pas techniquement correct. Tout ordinateur à usage général possède une unité centrale qui traite les commandes binaires spécifiées par les logiciels. Il est donc compréhensible que les gens confondent

processeur et CPU. Cependant, en plus d'une unité centrale, les ordinateurs modernes comprennent souvent d'autres processeurs spécifiques aux tâches. Le processeur supplémentaire le plus reconnaissable est peut-être le GPU (*Graphical Processing Unit*). Ainsi, si une CPU est un processeur, tous les processeurs ne sont pas des CPU.

Pour la plupart des gens, l'architecture du CPU est une référence aux instructions que le processeur prend en charge. Bien qu'Intel et AMD fabriquent des processeurs prenant en charge les mêmes instructions, il est important de faire la distinction entre les différents fournisseurs en raison des différences de conditionnement, de performances et de consommation d'énergie qui leur sont propres. Les distributions de logiciels utilisent couramment ces désignations pour spécifier l'ensemble minimum d'instructions dont ils ont besoin pour fonctionner :

### **i386**

Fait référence au jeu d'instructions 32 bits associé à l'Intel 80386.

### **x86**

Fait généralement référence aux jeux d'instructions 32 bits associés aux successeurs du 80386, tels que le 80486, le 80586 et le Pentium.

### **x64 / x86-64**

Processeurs de référence qui prennent en charge les instructions 32 bits et 64 bits de la famille x86.

### **AMD**

Une référence au support x86 par les processeurs AMD.

### **AMD64**

Une référence au support x64 par les processeurs AMD.

### **ARM**

Fait référence à un processeur RISC (*Reduced Instruction Set Computer*) qui n'est pas basé sur le jeu d'instructions x86. Communément utilisé par les périphériques embarqués, mobiles, sur tablette et sur batterie. Une version de Linux pour ARM est utilisée par le Raspberry Pi.

Le fichier `/proc/cpuinfo` contient des informations détaillées sur le processeur d'un système. Malheureusement, ces informations ne sont pas accessibles à tous les utilisateurs. Un résultat plus général peut être obtenu avec la commande `lscpu`. Sortie d'un Raspberry Pi B+ :

```
$ lscpu
Architecture:          armv7l
Byte Order:            Little Endian
```

```
CPU(s): 4
On-line CPU(s) list: 0-3
Thread(s) per core: 1
Core(s) per socket: 4
Socket(s): 1
Model: 4
Model name: ARMv7 Processor rev 4 (v7l)
CPU max MHz: 1400.0000
CPU min MHz: 600.0000
BogoMIPS: 38.40
Flags: half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32
lpae evtstrm crc32
```

Pour la plupart des gens, la myriade de fournisseurs, de familles de processeurs et de facteurs de spécification représente un éventail de choix déconcertant. Quoi qu'il en soit, il existe plusieurs facteurs associés aux CPUs et aux processeurs que même les utilisateurs et les administrateurs généraux doivent souvent prendre en compte lorsqu'ils doivent spécifier des environnements opérationnels :

### Taille des bits

Pour les processeurs, ce nombre est fonction à la fois de la taille native des données qu'ils manipulent et de la quantité de mémoire à laquelle ils peuvent accéder. La plupart des systèmes modernes sont soit 32 bits, soit 64 bits. Si une application a besoin d'accéder à plus de 4 gigaoctets de mémoire, elle doit fonctionner sur un système 64 bits, car 4 gigaoctets est l'adresse maximale qui peut être représentée en utilisant 32 bits. Et, alors que les applications 32 bits peuvent généralement s'exécuter sur des systèmes 64 bits, les applications 64 bits ne peuvent pas s'exécuter sur des systèmes 32 bits.

### Vitesse de l'horloge

Souvent exprimé en mégahertz (MHz) ou gigahertz (GHz). Il s'agit de la vitesse à laquelle un processeur traite les instructions. Mais la vitesse du processeur n'est qu'un des facteurs qui influent sur les temps de réponse du système, les temps d'attente et le débit. Même un utilisateur multitâche actif garde rarement le processeur d'un PC de bureau commun actif plus de 2 ou 3 % du temps. Quoi qu'il en soit, si vous utilisez fréquemment des applications à forte intensité de calcul impliquant des activités telles que le chiffrement ou le rendu vidéo, alors la vitesse du processeur peut avoir un impact significatif sur le débit et le temps d'attente.

### Le cache

Pour fonctionner, les processeurs ont besoin d'un flux constant d'instructions et de données. Le coût et la consommation d'énergie d'une mémoire système de plusieurs gigaoctets accessible à la vitesse d'horloge de l'unité centrale seraient prohibitifs. La mémoire cache à la vitesse du



CPU est intégrée à la puce du CPU pour fournir un tampon à grande vitesse entre les CPU et la mémoire système. La mémoire cache est séparée en plusieurs couches, communément appelées L1, L2, L3 et même L4. Dans le cas de la mémoire cache, plus est souvent mieux.

## Les noyaux

Le terme "noyau" fait référence à une unité centrale individuelle. En plus de représenter une unité centrale physique, la *technologie Hyper-Threading* (HTT) permet à une seule unité centrale physique de traiter simultanément plusieurs instructions, agissant ainsi virtuellement comme plusieurs unités centrales physiques. Le plus souvent, plusieurs cœurs physiques sont intégrés dans une seule puce de processeur physique. Toutefois, certaines cartes mères prennent en charge plusieurs puces de processeur physique. En théorie, le fait d'avoir plus de cœurs pour traiter les tâches semblerait toujours donner un meilleur rendement du système. Malheureusement, les applications de bureau ne tiennent souvent les CPU occupés que 2 ou 3 % du temps, de sorte que l'ajout de CPU, le plus souvent inactifs, n'améliorera probablement que très peu le débit. Un plus grand nombre de cœurs est mieux adapté à l'exécution d'applications écrites pour avoir plusieurs threads indépendants de fonctionnement, tels que le rendu d'images vidéo, le rendu de pages web ou les environnements de machines virtuelles multi-utilisateurs.

## Le stockage

Les dispositifs de stockage offrent une méthode pour conserver les programmes et les données. Les *disques durs* (HDD) et les *disques durs à semi-conducteurs* (SSD) sont les dispositifs de stockage les plus courants pour les serveurs et les ordinateurs de bureau. Les clés USB et les dispositifs optiques tels que les DVD sont également utilisés, mais rarement comme dispositif principal.

Comme son nom l'indique, un disque dur stocke des informations sur un ou plusieurs disques physiques rigides. Les disques physiques sont recouverts d'un support magnétique pour permettre le stockage. Les disques sont contenus dans un boîtier scellé car la poussière, les petites particules et même les empreintes digitales interféreraient avec la capacité du disque dur à lire et à écrire sur le support magnétique.

Les disques SSD sont en fait des versions plus sophistiquées des clés USB avec une capacité nettement plus importante. Les disques SSD stockent des informations dans des micropuces, ce qui fait qu'il n'y a pas de pièces mobiles.

Bien que les technologies sous-jacentes des disques HDD et des disques SSD soient différentes, des facteurs importants peuvent être comparés. La capacité des disques HDD est basée sur la mise à l'échelle des composants physiques tandis que la capacité des disques SSD dépend du nombre de micro-puces. Par gigaoctet, les disques SSD coûtent entre 3 et 10 fois le coût d'un disque HDD. Pour lire ou écrire, un disque HDD doit attendre qu'un emplacement sur un disque tourne vers un

emplacement connu alors que les disques SSD sont à accès aléatoire. Les vitesses d'accès aux disques SSD sont généralement de 3 à 5 fois plus rapides que celles des disques HDD. Comme ils ne comportent pas de pièces mobiles, les SSD consomment moins d'énergie et sont plus fiables que les disques HDD.

La capacité de stockage est en constante augmentation pour les disques HDD et les disques SSD. Aujourd'hui, on trouve couramment des disques HDD de 5 téraoctets et des disques SSD de 1 téraoctet. Quoi qu'il en soit, une grande capacité de stockage n'est pas toujours mieux. Lorsqu'un périphérique de stockage tombe en panne, les informations qu'il contenait ne sont plus disponibles. Et bien sûr, la sauvegarde prend plus de temps lorsqu'il y a plus d'informations à sauvegarder. Pour les applications qui lisent et écrivent beaucoup de données, la latence et les performances peuvent être plus importantes que la capacité.

Les systèmes modernes utilisent le SCSI (*Small Computer System Interface*) ou le SATA (*Serial AT Attachment*) pour se connecter aux périphériques de stockage. Ces interfaces sont généralement prises en charge par le connecteur approprié sur la carte mère. La charge initiale provient d'un périphérique de stockage connecté à la carte mère. Les paramètres du firmware définissent l'ordre dans lequel les périphériques sont accédés pour ce chargement initial.

Les systèmes de stockage connus sous le nom de RAID (*Redundant Array of Independent Disks*) sont une implémentation courante pour éviter la perte d'informations. Une matrice RAID est constituée de plusieurs dispositifs physiques contenant des copies d'informations. Si l'un des dispositifs tombe en panne, toutes les informations sont encore disponibles. Les différentes configurations physiques de RAID sont référencées comme 0, 1, 5, 6 et 10. Chaque désignation a une taille de stockage, des caractéristiques de performance et des façons de stocker des données redondantes ou des sommes de contrôle différentes pour la récupération des données. Au-delà d'une certaine surcharge de configuration administrative, l'existence du RAID est effectivement transparente pour les utilisateurs.

Les périphériques de stockage lisent et écrivent généralement les données sous forme de blocs d'octets. La commande `lsblk` peut être utilisée pour lister des blocs de périphériques disponibles pour un système. L'exemple suivant a été exécuté sur un Raspberry Pi en utilisant une carte SD comme périphérique de stockage. Les détails de la sortie sont couverts par des informations dans les leçons sur les *partitions* et les *pilotes* qui suivent :

```
$ lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0       179:0    0 29.7G  0 disk
+-mmcblk0p1   179:1    0 43.9M  0 part /boot
+-mmcblk0p2   179:2    0 29.7G  0 part /
```

## Les partitions

Un dispositif de stockage est en fait une longue séquence de lieux de stockage. Le partitionnement est le mécanisme qui indique à Linux s'il doit voir ces emplacements de stockage comme une seule séquence ou plusieurs séquences indépendantes. Chaque partition est traitée comme s'il s'agissait d'un dispositif individuel. La plupart du temps, les partitions sont créées lors de la première configuration d'un système. Si un changement est nécessaire, des outils administratifs sont disponibles pour gérer le partitionnement des périphériques.

Alors pourquoi serait-il souhaitable d'avoir plusieurs partitions ? La gestion du stockage disponible, l'isolation des charges de chiffrement ou la prise en charge de plusieurs systèmes de fichiers sont des exemples d'utilisation de partitions. Les partitions permettent d'avoir un seul périphérique de stockage qui peut démarrer sous différents systèmes d'exploitation.

Alors que Linux peut reconnaître la séquence de stockage d'un périphérique brut, un périphérique brut ne peut pas être utilisé tel quel. Pour utiliser un périphérique brut, il doit être formaté. Le formatage écrit un système de fichiers sur un périphérique et le prépare pour les opérations de fichiers. Sans système de fichiers, un périphérique ne peut pas être utilisé pour des opérations liées aux fichiers.

Les utilisateurs voient les partitions comme s'il s'agissait de dispositifs individuels. Il est donc facile de ne pas voir qu'il s'agit toujours d'un seul périphérique physique. En particulier, les opérations de périphérique à périphérique qui sont en fait des opérations de partition à partition n'auront pas les performances attendues. Un périphérique unique est un mécanisme physique avec un ensemble de matériel de lecture/écriture. Plus important encore, vous ne pouvez pas utiliser les partitions d'un seul périphérique physique comme une conception tolérante aux pannes. Si le périphérique tombe en panne, toutes les partitions tombent en panne et il n'y a donc pas de tolérance aux pannes.

### NOTE

*Logical Volume Manager* (LVM) est une fonctionnalité logicielle qui permet aux administrateurs de combiner des disques et des partitions de disque individuels et de les traiter comme s'il s'agissait d'un seul et même disque.

## Les périphériques

Les serveurs et les postes de travail ont besoin d'une combinaison de CPU, de mémoire système et de stockage pour fonctionner. Mais ces composants fondamentaux n'ont pas d'interface directe avec le monde extérieur. Les périphériques sont les dispositifs qui fournissent aux systèmes les entrées, les sorties et l'accès au reste du monde réel.

La plupart des cartes mères sont équipées de connecteurs externes intégrés et prennent en charge

les firmwares des interfaces de périphériques hérités courants, tels que le clavier, la souris, le son, la vidéo et le réseau. Les cartes mères récentes sont généralement dotées d'un connecteur Ethernet pour la prise en charge des réseaux, d'un connecteur HDMI pour les besoins graphiques de base et d'un ou plusieurs connecteurs USB (*Universal Serial Bus*) pour la plupart des autres fonctions. Il existe plusieurs versions d'USB avec des vitesses et des caractéristiques physiques différentes. Plusieurs versions de ports USB sont communes sur une même carte mère.

Les cartes mères peuvent également avoir un ou plusieurs emplacements d'extension (slots). Les emplacements d'extension permettent aux utilisateurs d'ajouter des cartes de circuit imprimé spéciales, appelées cartes d'extension, qui prennent en charge des périphériques personnalisés, anciens et non standard. Les interfaces graphiques, sonores et réseau sont des cartes d'extension courantes. Les cartes d'extension prennent également en charge le RAID et les interfaces anciennes de format spécial impliquant des connexions série et parallèle.

Les configurations de *système sur puce* (SoC : *System on a Chip*) offrent des avantages en termes de puissance, de performance, d'espace et de fiabilité par rapport aux configurations de carte mère en regroupant les processeurs, la mémoire système, les disques SSD et le matériel pour contrôler les périphériques dans un seul boîtier de circuit intégré. Les périphériques pris en charge par les configurations SoC sont limités par les composants embarqués. Ainsi, les configurations SoC ont tendance à être développées pour des utilisations spécifiques. Les téléphones, les tablettes et autres appareils portables sont souvent basés sur la technologie SoC.

Certains systèmes intègrent des périphériques. Les ordinateurs portables sont similaires aux stations de travail mais intègrent par défaut des périphériques d'affichage, de clavier et de souris. Les systèmes tout-en-un sont similaires aux ordinateurs portables mais nécessitent des périphériques de souris et de clavier. Les contrôleurs basés sur une carte mère ou un SoC sont souvent équipés de périphériques intégrés adaptés à une utilisation spécifique.

## Les Pilotes et Fichiers de Périphériques

Jusqu'à présent, cette leçon a présenté des informations sur les processeurs, la mémoire, les disques, le partitionnement, le formatage et les périphériques. Mais exiger des utilisateurs généraux qu'ils traitent les détails spécifiques de chacun des périphériques de leur système rendrait ces systèmes inutilisables. De même, les développeurs de logiciels devraient modifier leur code pour chaque périphérique nouveau ou modifié qu'ils doivent prendre en charge.

La solution à ce problème de "traitement des détails" est fournie par le pilote de périphérique. Les pilotes de périphériques acceptent un ensemble standard de demandes puis traduisent ces demandes en activités de contrôle appropriées au périphérique. Les pilotes de périphériques sont ce qui vous permet, à vous et aux applications que vous exécutez, de lire à partir du fichier `/home/carol/stuff` sans vous soucier de savoir si ce fichier se trouve sur un disque HDD, un

disque SSD, une clé USB, un stockage chiffré ou tout autre périphérique.

Les fichiers de périphériques se trouvent dans le répertoire `/dev` et identifient les périphériques physiques, les accès aux périphériques et les pilotes pris en charge. Par convention, dans les systèmes modernes utilisant des périphériques de stockage SCSI ou SATA, le nom du fichier de spécification commence par le préfixe `sd`. Le préfixe est suivi d'une lettre telle que `a` ou `b` indiquant un périphérique physique. Après le préfixe et l'identificateur du périphérique vient un numéro indiquant une partition dans le périphérique physique. Ainsi, `/dev/sda` fait référence à la totalité du premier périphérique de stockage tandis que `/dev/sda3` fait référence à la partition 3 du premier périphérique de stockage. Le fichier de périphérique contient, pour chaque type de périphérique une convention de nommage appropriée. Bien que la couverture de toutes les conventions de nommage possibles dépasse le cadre de cette leçon, il est important de se rappeler que ces conventions sont essentielles pour rendre possible l'administration du système.

Bien que le contenu du répertoire `/dev` dépasse la portée de cette leçon, il est instructif de regarder l'entrée d'un périphérique de stockage. Les fichiers de périphériques pour les cartes SD utilisent généralement `mmcblk` comme préfixe :

```
$ ls -l mmcblk*
brw-rw---- 1 root disk 179, 0 Jun 30 01:17 mmcblk0
brw-rw---- 1 root disk 179, 1 Jun 30 01:17 mmcblk0p1
brw-rw---- 1 root disk 179, 2 Jun 30 01:17 mmcblk0p2
```

Les détails du listage pour un fichier de périphérique sont différents des détails de fichier typiques :

- Contrairement à un fichier ou à un répertoire, la première lettre du champ d'autorisation est `b`. Cela indique que les blocs sont lus et écrits sur le périphérique en blocs plutôt qu'en caractères individuels.
- Le champ de taille est constitué de deux valeurs séparées par une virgule plutôt que d'une valeur unique. La première valeur indique généralement un pilote particulier au sein du noyau et la seconde valeur spécifie un périphérique spécifique géré par le pilote.
- Le nom du fichier utilise un numéro pour le dispositif physique, de sorte que la convention d'appellation s'adapte en spécifiant le suffixe de partition sous la forme d'un `p` suivi d'un chiffre.

#### NOTE

Chaque périphérique du système doit avoir une entrée dans `/dev`. Comme le contenu du répertoire `/dev` est créé lors de l'installation, il y a souvent des entrées pour chaque pilote et périphérique possible, même si aucun périphérique physique n'existe.

## Exercices Guidés

1. Décrivez ces termes :

Processeur	
CPU	
GPU	

2. Si vous utilisez principalement des applications de montage vidéo (une activité à forte intensité de calcul), quels sont les composants et les caractéristiques qui devraient avoir le plus d'impact sur la convivialité du système :

Les cœurs de CPU	
La vitesse du CPU	
La mémoire système disponible	
Le système de stockage	
GPU	
L'affichage vidéo	
Aucune de ces réponses	

3. Quel devrait être le nom du fichier de périphérique dans `/dev` pour la partition 3 du troisième lecteur SATA d'un système ?

<code>sd3p3</code>	
<code>sdcp3</code>	
<code>sdc3</code>	
Aucune de ces réponses	

## Exercices d'Exploration

1. Exécutez la commande `lsblk` sur votre système. Identifiez les paramètres ci-dessous. Si un système n'est pas immédiatement disponible, considérez la sortie de `lsblk -f` pour le système Raspberry Pi mentionné dans la section "Stockage" ci-dessus :

```
$ lsblk -f
NAME                FSTYPE LABEL  UUID                                  MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat    boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4     rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

- Le type de périphériques et leur nombre
- La structure de partition de chaque périphérique
- Le type de système de fichiers et de support pour chaque partition

## Résumé

Un système est la somme de ses composants. Les différents composants ont un impact différent sur le coût, les performances et la facilité d'utilisation. S'il existe des configurations communes pour les stations de travail et les serveurs, il n'y a pas de configuration optimale unique.



# Réponses aux Exercices Guidés

## 1. Décrivez ces termes :

### **Processeur**

Terme général qui s'applique à tout type de processeur. Souvent utilisé à tort comme synonyme de CPU.

### **CPU**

Une unité centrale de traitement. Une unité de traitement fournissant un soutien pour les tâches informatiques générales.

### **GPU**

Une unité de traitement graphique. Une unité de traitement optimisée pour soutenir les activités liées à la présentation des graphiques.

## 2. Si vous utilisez principalement des applications de montage vidéo (une activité à forte intensité de calcul), quels sont les composants et les caractéristiques qui devraient avoir le plus d'impact sur la convivialité du système :

### **Les cœurs de CPU**

Oui. Les cœurs multiples prennent en charge les tâches de présentation et de rendu simultanées requises par le montage vidéo.

### **La vitesse du CPU**

Oui. Le rendu vidéo nécessite une quantité importante d'activités de calcul.

### **La mémoire système disponible**

Probablement. La vidéo non compressée utilisée pour le montage est volumineuse. Les systèmes d'usage général sont souvent dotés de 8 gigaoctets de mémoire. Une mémoire de 16 ou même 32 gigaoctets permet au système de gérer un plus grand nombre d'images vidéo non compressées, ce qui rend les activités de montage plus efficaces.

### **Le système de stockage**

Oui. Les fichiers vidéo sont volumineux. La surcharge des lecteurs SSD locaux permet un transfert plus efficace. Les disques réseau plus lents risquent d'être contre-productifs.

### **Le GPU**

Non. Le GPU influe principalement sur la présentation de la vidéo rendue.

### L'affichage vidéo

Non. L'affichage de la vidéo a principalement un impact sur la présentation de la vidéo rendue.

### Aucune de ces réponses

Non. Certains de ces facteurs ont des répercussions évidentes sur la facilité d'utilisation de votre système.

3. Quel devrait être le nom du fichier de périphérique dans `/dev` pour la partition 3 du troisième lecteur SATA d'un système ?

sd3p3	Incorrect. Le lecteur 3 serait <code>sd3</code> et non <code>sd3</code>
sdcp3	Incorrect. La partition 3 serait <code>3</code> et non pas <code>p3</code>
sd3	Correct
None of the above	Ce n'est pas correct. La bonne réponse est l'un des choix.

# Réponses aux Exercices d'Exploration

1. Exécutez la commande `lsblk` sur votre système. Identifiez les paramètres ci-dessous. Si un système n'est pas immédiatement disponible, considérez la sortie de `lsblk -f` pour le système Raspberry Pi mentionné dans la section "Stockage" ci-dessus :

```
$ lsblk -f
NAME          FSTYPE LABEL  UUID                                  MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat   boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4    rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

Les réponses qui suivent sont basées sur le listage `lsblk -f` pour le système Raspberry Pi ci-dessus. Vos réponses peuvent varier :

## Le type de périphériques et leur nombre

Il existe un dispositif : `mmcblk0`. Vous savez par convention que le `mmcblk` serait une carte mémoire SD.

## La structure de partition de chaque périphérique

Il y a deux partitions : `mmcblk0p1` et `mmcblk0p2`.

## Le type de système de fichiers et de support pour chaque partition

La partition 1 utilise le système de fichiers `vfat`. Il est utilisé pour démarrer le système et est monté en tant que `/boot`. La partition 2 utilise le système de fichiers `ext4`. Il est utilisé comme système de fichiers principal et est monté sous la forme `/`.



**Linux  
Professional  
Institute**

## 4.3 Localisation des données

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 4.3

### Valeur

3

### Domaines de connaissance les plus importants

- Programmes et configuration
- Processus
- Adresses en mémoire
- Messagerie système
- Journalisation

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `ps`, `top`, `free`
- `syslog`, `dmesg`
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`



## 4.3 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	4 Le Système d'Exploitation Linux
<b>Objectif :</b>	4.3 Lieux de Stockage des Données
<b>Leçon:</b>	1 sur 2

### Introduction

Pour un système d'exploitation, tout est considéré comme des données. Pour Linux, tout est considéré comme un fichier : programmes, fichiers réguliers, répertoires, périphériques à blocs (disques durs, etc.), périphériques à caractères (consoles, etc.), processus du noyau, sockets, partitions, liens, etc. La structure des répertoires de Linux, en partant de la *racine* `/`, est un ensemble de fichiers contenant des données. Le fait que tout soit un fichier est une caractéristique puissante de Linux car elle permet de modifier pratiquement chaque aspect du système.

Dans cette leçon, nous discuterons des différents emplacements dans lesquels les données importantes sont stockées selon la norme de hiérarchie des systèmes de fichiers Linux (FHS). Certains de ces emplacements sont de véritables répertoires qui stockent les données de manière persistante sur les disques, tandis que d'autres sont des pseudo-systèmes de fichiers chargés en mémoire qui nous donnent accès aux données du sous-système du noyau telles que les processus en cours d'exécution, l'utilisation de la mémoire, la configuration matérielle, etc. Les données stockées dans ces répertoires virtuels sont utilisées par un certain nombre de commandes qui nous permettent de les surveiller et de les manipuler.

## Les Programmes et leurs Configurations

Les données importantes sur un système Linux sont sans aucun doute les programmes et leurs fichiers de configuration. Les premiers sont des fichiers exécutables contenant des ensembles d'instructions à exécuter par le processeur de l'ordinateur, tandis que les seconds sont généralement des documents texte qui contrôlent le fonctionnement d'un programme. Les fichiers exécutables peuvent être soit des fichiers binaires, soit des fichiers texte. Les fichiers texte exécutables sont appelés scripts. Les données de configuration sous Linux sont traditionnellement stockées dans des fichiers texte également, bien qu'il existe différents styles de représentation des données de configuration.

### Lieux de Stockage des Fichiers Binaires

Comme tout autre fichier, les fichiers exécutables se trouvent dans des répertoires reliés à /. Plus précisément, les programmes sont distribués selon une structure à trois niveaux : le premier niveau (/) comprend les programmes qui peuvent être nécessaires en mode monoposte, le deuxième niveau (/usr) contient la plupart des programmes multi-utilisateurs et le troisième niveau (/usr/local) est utilisé pour stocker les logiciels qui ne sont pas fournis par la distribution et qui ont été compilés localement.

Les lieux typiques pour les programmes comprennent :

#### **/sbin**

Il contient des binaires essentiels pour l'administration du système, tels que `parted` ou `ip`.

#### **/bin**

Il contient des binaires essentiels pour tous les utilisateurs tels que `ls`, `mv`, ou `mkdir`.

#### **/usr/sbin**

Il stocke des binaires pour l'administration du système tels que `deluser`, ou `groupadd`.

#### **/usr/bin**

Il contient la plupart des fichiers exécutables comme : `free`, `pstree`, `sudo` ou `man` qui peuvent être utilisés par tous les utilisateurs.

#### **/usr/local/sbin**

Il est utilisé pour stocker les programmes installés localement pour l'administration du système qui ne sont pas gérés par le gestionnaire de paquets du système.

#### **/usr/local/bin**

Il sert le même objectif que `/usr/local/sbin` mais pour les programmes d'utilisateurs

réguliers.

Récemment, certaines distributions ont commencé à remplacer `/bin` et `/sbin` par des liens symboliques vers `/usr/bin` et `/usr/sbin`.

**NOTE**

Le répertoire `/opt` est parfois utilisé pour stocker des applications tierces facultatives.

En dehors de ces répertoires, les utilisateurs réguliers peuvent avoir leurs propres programmes dans l'un ou l'autre des répertoires :

- `/home/$USER/bin`
- `/home/$USER/.local/bin`

Vous pouvez découvrir quels sont les répertoires disponibles pour exécuter des binaires en affichant la variable `PATH` avec `echo $PATH`. Pour plus d'informations sur `PATH`, consultez les leçons sur les variables et la personnalisation du shell.

On peut trouver l'emplacement des programmes avec la commande `which` :

```
$ which git
/usr/bin/git
```

## Lieux de Stockage des Fichiers de Configuration

### Le répertoire `/etc`

Dans les premiers temps d'Unix, il y avait un dossier pour chaque type de données, comme `/bin` pour les binaires et `/boot` pour le(s) noyau(x). Cependant, `/etc` (signifiant *et cetera*) a été créé comme un répertoire fourre-tout pour stocker tous les fichiers qui n'appartenaient pas aux autres catégories. La plupart de ces fichiers étaient des fichiers de configuration. Au fil du temps, de plus en plus de fichiers de configuration ont été ajoutés, de sorte que `/etc` est devenu le dossier principal pour les fichiers de configuration des programmes. Comme indiqué ci-dessus, un fichier de configuration est généralement un fichier local en texte brut (par opposition à un fichier binaire) qui contrôle le fonctionnement d'un programme.

Dans `/etc`, nous pouvons trouver différents modèles pour les noms des fichiers de configuration :

- Fichiers avec une extension *ad hoc* ou sans extension du tout, par exemple

**group**

Base de données des groupes du système.

**hostname**

Nom d'hôte de l'ordinateur.

**hosts**

Liste des adresses IP et traduction de leur nom d'hôte.

**passwd**

Base de données des utilisateurs du système composée de sept champs séparés par des deux points fournissant des informations sur les utilisateurs.

**profile**

Fichier de configuration de Bash pour l'ensemble du système.

**shadow**

Fichier chiffré pour les mots de passe des utilisateurs.

- Fichiers d'initialisation se terminant par **rc** :

**bash.bashrc**

Fichier **.bashrc** à l'échelle du système pour les shells bash interactifs.

**nanorc**

Exemple de fichier d'initialisation pour GNU nano (un simple éditeur de texte qui est normalement livré avec toutes les distributions).

- Les fichiers se terminant par **.conf** :

**resolv.conf**

Le fichier de configuration du résolveur qui permet d'accéder au système de noms de domaine Internet (DNS).

**sysctl.conf**

Fichier de configuration pour définir les variables système du noyau.

- Répertoires avec le suffixe **.d** :

Certains programmes avec un fichier de configuration unique (**\*.conf** ou autre) ont évolué pour avoir un répertoire **\*.d** dédié qui permet de construire des configurations modulaires et



plus robustes. Par exemple, pour configurer logrotate, vous trouverez `logrotate.conf`, mais aussi le répertoire `logrotate.d`.

Cette approche est utile dans les cas où différentes applications nécessitent des configurations pour le même service spécifique. Si, par exemple, un progiciel de serveur web contient une configuration logrotate, cette configuration peut maintenant être placée dans un fichier dédié dans le répertoire `logrotate.d`. Ce fichier peut être mis à jour par le progiciel de serveur web sans interférer avec la configuration logrotate restante. De même, les paquets peuvent ajouter des tâches spécifiques en plaçant des fichiers dans le répertoire `/etc/cron.d` au lieu de modifier `/etc/crontab`.

Dans Debian et les dérivées de Debian une telle approche a été appliquée à la liste des sources fiables lues par l'outil de gestion des paquets `apt` : outre le classique `/etc/apt/sources.list`, on trouve maintenant le répertoire `/etc/apt/sources.list.d` :

```
$ ls /etc/apt/sources*  
/etc/apt/sources.list  
/etc/apt/sources.list.d:
```

## Les Fichiers de Configuration dans HOME (Dotfiles)

Au niveau de l'utilisateur, les programmes stockent leurs configurations et paramètres dans des fichiers cachés dans le répertoire personnel de l'utilisateur (également représenté par `~`). N'oubliez pas que les fichiers cachés commencent par un point ( `.` ) d'où leur nom : *dotfiles*.

Certains de ces dotfiles sont des scripts Bash qui personnalisent la session shell de l'utilisateur et sont générés dès que l'utilisateur se connecte au système :

### **.bash\_history**

Il stocke l'historique de la ligne de commande.

### **.bash\_logout**

Il comprend des commandes à exécuter lorsque l'on quitte le shell de connexion.

### **.bashrc**

Script d'initialisation de Bash pour les shells sans login.

### **.profile**

Le script d'initialisation de Bash pour les shells de connexion.

## **NOTE**

Reportez-vous à la leçon sur les “Bases de la Ligne de Commande” pour en savoir

plus sur Bash et ses fichiers d'initialisation.

Les fichiers de configuration d'autres programmes spécifiques à l'utilisateur sont obtenus au démarrage de leurs programmes respectifs : `.gitconfig`, `.emacs.d`, `.ssh`, etc.

## Le Noyau Linux

Avant qu'un processus puisse s'exécuter, le noyau doit être chargé dans une zone de mémoire protégée. Ensuite, le processus avec le PID 1 (le plus souvent `systemd` de nos jours) déclenche la chaîne de processus, c'est-à-dire qu'un processus en démarre un ou plusieurs autres et ainsi de suite. Une fois les processus actifs, le noyau Linux se charge de leur allouer des ressources (clavier, souris, disques, mémoire, interfaces réseau, etc.).

### NOTE

Avant `systemd`, `/sbin/init` était toujours le premier processus dans un système Linux dans le cadre du gestionnaire de système *System V Init*. En fait, on trouve encore `/sbin/init` actuellement mais il est lié à `/lib/systemd/systemd`.

## Lieux de Stockage des Noyaux : `/boot`

Le noyau réside dans `/boot` avec d'autres fichiers liés au démarrage. La plupart de ces fichiers comportent dans leur nom les composants du numéro de version du noyau (version du noyau, révision majeure, révision mineure et numéro de correctif).

Le répertoire `/boot` comprend les types de fichiers suivants, dont les noms correspondent à la version respective du noyau :

### `config-4.9.0-9-amd64`

Les paramètres de configuration du noyau tels que les options et les modules qui ont été compilés avec le noyau.

### `initrd.img-4.9.0-9-amd64`

Image initiale du disque RAM qui aide au processus de démarrage en chargeant un système de fichiers racine temporaire en mémoire.

### `System-map-4.9.0-9-amd64`

Le fichier `System-map` (sur certains systèmes, il sera nommé `System.map`) contient les emplacements des adresses mémoire pour les noms des symboles du noyau. Chaque fois qu'un noyau est reconstruit, le contenu du fichier change car les emplacements mémoire peuvent être différents. Le noyau utilise ce fichier pour rechercher les adresses mémoire d'un symbole de noyau particulier, ou vice-versa.

## **vmlinuz-4.9.0-9-amd64**

Le noyau proprement dit dans un format auto-extractible, peu encombrant et compressé (d'où le **z** dans **vmlinuz** ; **vm** signifie mémoire virtuelle et a commencé à être utilisé lorsque le noyau a obtenu la prise en charge de la mémoire virtuelle pour la première fois).

## **grub**

Répertoire de configuration pour le bootloader **grub2**.

Comme il s'agit d'une caractéristique essentielle du système d'exploitation, plus d'un noyau et ses fichiers associés sont conservés dans **/boot** au cas où le noyau par défaut deviendrait défectueux et que nous devions nous rabattre sur une version antérieure pour au moins être capable de démarrer le système et de le réparer.

## **Le Répertoire /proc**

Le répertoire **/proc** est l'un des systèmes de fichiers dits virtuels ou pseudo-virtuels puisque son contenu n'est pas écrit sur le disque, mais chargé en mémoire. Il est alimenté dynamiquement à chaque démarrage de l'ordinateur et reflète constamment l'état actuel du système. Le répertoire **/proc** contient des informations sur :

- Les processus en cours
- La configuration du noyau
- Le matériel du système

Outre toutes les données concernant les processus que nous verrons dans la prochaine leçon, ce répertoire contient également des fichiers contenant des informations sur le matériel du système et les paramètres de configuration du noyau. Certains de ces fichiers comprennent :

### **/proc/cpuinfo**

Il stocke des informations sur le CPU du système :

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 158
model name    : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping      : 10
cpu MHz       : 3696.000
```

```
cache size : 12288 KB
(...)
```

## **/proc/cmdline**

Il stocke les chaînes de caractères passées au noyau au démarrage :

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro
quiet
```

## **/proc/modules**

Il montre la liste des modules chargés dans le noyau :

```
$ cat /proc/modules
nls_utf8 16384 1 - Live 0xffffffffc0644000
isofs 40960 1 - Live 0xffffffffc0635000
udf 90112 0 - Live 0xffffffffc061e000
crc_itu_t 16384 1 udf, Live 0xffffffffc04be000
fuse 98304 3 - Live 0xffffffffc0605000
vboxsf 45056 0 - Live 0xffffffffc05f9000 (0)
joydev 20480 0 - Live 0xffffffffc056e000
vboxguest 327680 5 vboxsf, Live 0xffffffffc05a8000 (0)
hid_generic 16384 0 - Live 0xffffffffc0569000
(...)
```

## **Le Répertoire /proc/sys**

Ce répertoire comprend les paramètres de configuration du noyau dans des fichiers classés en catégories par sous-répertoire :

```
$ ls /proc/sys
abi debug dev fs kernel net user vm
```

La plupart de ces fichiers agissent comme un interrupteur et contiennent l'une des deux valeurs possibles : 0 ou 1 ("on" ou "off"). Par exemple :

### **/proc/sys/net/ipv4/ip\_forward**

Valeur qui permet à notre machine d'agir en tant que routeur (être capable de transmettre des paquets) ou qui le désactive :

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Il y a cependant quelques exceptions :

### **/proc/sys/kernel/pid\_max**

Le PID maximum autorisé :

```
$ cat /proc/sys/kernel/pid_max
32768
```

Soyez très prudent lorsque vous modifiez les paramètres du noyau, car une valeur incorrecte peut entraîner une instabilité du système.

## **Les Périphériques Matériels**

Rappelez-vous, sous Linux, “tout est un fichier”. Cela implique que les informations sur les périphériques matériels ainsi que les paramètres de configuration propres au noyau sont tous stockés dans des fichiers spéciaux qui résident dans des répertoires virtuels.

### **Le Répertoire /dev**

Le répertoire de *périphériques* /dev contient les fichiers de périphériques (ou nœuds) de tous les périphériques matériels connectés. Ces fichiers de périphériques sont utilisés comme interface entre les périphériques et les processus qui les utilisent. Chaque fichier de périphérique appartient à l'une des deux catégories suivantes :

#### **Périphériques à blocs**

Ce sont ceux dans lesquels les données sont lues et écrites en blocs qui peuvent être adressés individuellement. Les exemples incluent les disques durs (et leurs partitions, comme /dev/sda1), les lecteurs flash USB, les CD, les DVD, etc.

#### **Périphérique à caractères**

Ce sont ceux dans lesquels les données sont lues et écrites séquentiellement, un caractère à la fois. Les exemples comprennent les claviers, la console de texte (/dev/console), les ports série (tels que /dev/ttyS0 et ainsi de suite), etc.

Lorsque vous répertoriez les fichiers de périphériques, assurez-vous d'utiliser `ls` avec l'option `-l`

pour faire la différence entre les deux. Nous pouvons par exemple rechercher les disques durs et les partitions :

```
# ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 may 25 17:02 /dev/sda
brw-rw---- 1 root disk 8, 1 may 25 17:02 /dev/sda1
brw-rw---- 1 root disk 8, 2 may 25 17:02 /dev/sda2
(...)
```

Ou pour les terminaux séries (TeleTYpewriter) :

```
# ls -l /dev/tty*
crw-rw-rw- 1 root tty      5, 0 may 25 17:26 /dev/tty
crw--w---- 1 root tty      4, 0 may 25 17:26 /dev/tty0
crw--w---- 1 root tty      4, 1 may 25 17:26 /dev/tty1
(...)
```

Remarquez que le premier caractère est **b** pour les périphériques à bloc et **c** pour les périphériques à caractères.

L'astérisque (\*) est un caractère de globbing qui signifie 0 ou plus de caractères. D'où son importance dans les commandes `ls -l /dev/sd*` et `ls -l /dev/tty*` ci-dessus. Pour en savoir plus sur ces caractères spéciaux, reportez-vous à la leçon sur le globbing.

En outre, `/dev` comprends des fichiers spéciaux qui sont très utiles pour différents objectifs de programmation :

### **`/dev/zero`**

Il fournit autant de caractères nuls que nécessaire.

### **`/dev/null`**

Aka *bit bucket*. Il rejette toutes les informations qui lui sont envoyées.

### **`/dev/urandom`**

Il génère des nombres pseudo-aléatoires.

## **Le Répertoire `/sys`**

Le système de fichiers `sys` (`sysfs`) est monté sur `/sys`. Il a été introduit avec l'arrivée du noyau 2.6

et a représenté une grande amélioration de `/proc/sys`.

Les processus doivent interagir avec les périphériques dans `/dev` et le noyau a donc besoin d'un répertoire qui contient des informations sur ces périphériques matériels. Ce répertoire est `/sys` et ses données sont classées par catégories. Par exemple, pour vérifier l'adresse MAC de votre carte réseau (`enp0s3`), vous affichez avec `cat` le fichier suivant :

```
$ cat /sys/class/net/enp0s3/address
08:00:27:02:b2:74
```

## Mémoire et Types de Mémoire

Généralement, pour qu'un programme commence à fonctionner, il doit être chargé en mémoire. Dans l'ensemble, lorsque nous parlons de mémoire, nous faisons référence à la *mémoire vive* (RAM), et quand on la compare au disque dur mécanique, elle a l'avantage d'être beaucoup plus rapide. D'un autre côté, elle est volatile (c'est-à-dire qu'une fois que l'ordinateur est éteint, les données disparaissent).

Malgré ce que l'on vient de mentionner, quand il s'agit de mémoire, on peut différencier deux types principaux dans un système Linux :

### La mémoire physique

Aussi appelée *RAM*, elle se présente sous la forme de puces composées de circuits intégrés contenant des millions de transistors et de condensateurs. Ces derniers forment à leur tour des cellules de mémoire (le bloc de base de la mémoire d'un ordinateur). À chacune de ces cellules est associée une adresse mémoire code en hexadécimale à laquelle on peut se référer en cas de besoin.

### Le Swap

Également appelé *espace d'échange*, c'est la partie de la mémoire virtuelle qui vit sur le disque dur et qui est utilisée lorsqu'il n'y a plus de mémoire vive disponible.

D'autre part, il y a le concept de *mémoire virtuelle* qui est une abstraction de la quantité totale de mémoire d'adressage utilisable (RAM, mais aussi espace disque) telle que vue par les applications.

`free` analyse `/proc/meminfo` et affiche la quantité de mémoire libre et utilisée dans le système de manière très claire :

```
$ free
```

	total	used	free	shared	buff/cache	available
--	-------	------	------	--------	------------	-----------

Mem:	4050960	1474960	1482260	96900	1093740	2246372
Swap:	4192252	0	4192252			

Expliquons les différentes colonnes :

### **total**

Quantité totale de mémoire physique et de mémoire Swap installée.

### **used**

Quantité de mémoire physique et de mémoire Swap actuellement utilisée.

### **free**

Quantité de mémoire physique et de mémoire Swap actuellement non utilisée.

### **shared**

Quantité de mémoire physique utilisée, principalement par `tmpfs`.

### **buff/cache**

Quantité de mémoire physique actuellement utilisée par les tampons du noyau et le cache des pages et les plaques.

### **available**

Estimation de la quantité de mémoire physique disponible pour les nouveaux processus.

Par défaut, `free` affiche les valeurs en kibi octets, mais permet à une variété d'options d'afficher ses résultats dans différentes unités de mesure. Voici quelques-unes de ses options :

### **-b**

Octets.

### **-m**

Mébi octets.

### **-g**

Gibi octets.

### **-h**

Format lisible par l'homme.

`-h` est toujours agréable à lire :



**\$ free -h**

	total	used	free	shared	buff/cache	available
Mem:	3,9G	1,4G	1,5G	75M	1,0G	2,2G
Swap:	4,0G	0B	4,0G			

**NOTE**

Un kibioctet (Kib) équivaut à 1024 octets tandis qu'un kilooctet (Ko) équivaut à 1 000 octets. Il en va de même respectivement pour les mébioctets, gibioctets, etc.

## Exercices Guidés

1. Utilisez la commande `which` pour connaître l'emplacement des programmes suivants et compléter le tableau :

Programme	Commande <code>which</code>	Chemin vers l'exécutable (sortie)	L'utilisateur a besoin de privilèges de root ?
<code>swapon</code>			
<code>kill</code>			
<code>cut</code>			
<code>usermod</code>			
<code>cron</code>			
<code>ps</code>			

2. Où se trouvent les fichiers suivants ?

Fichier	/etc	~
<code>.bashrc</code>		
<code>bash.bashrc</code>		
<code>passwd</code>		
<code>.profile</code>		
<code>resolv.conf</code>		
<code>sysctl.conf</code>		

3. Expliquez la signification des éléments numériques du fichier de noyau `vmlinux-4.15.0-50-generic` trouvé dans `/boot` :

Élément de numéro	Signification
4	
15	
0	
50	

4. Quelle commande utiliseriez-vous pour lister tous les disques durs et les partitions dans `/dev` ?

## Exercices d'Exploration

1. Les fichiers de périphériques pour les disques durs sont représentés sur la base des contrôleurs qu'ils utilisent comme nous l'avons vu `/dev/sd*` pour les lecteurs utilisant SCSI (Small Computer System Interface) et SATA (Serial Advanced Technology Attachment), mais
  - Comment les anciens lecteurs IDE (Integrated Drive Electronics) étaient-ils représentés ?
  - Et les lecteurs modernes NVMe (Non-Volatile Memory Express) ?
2. Consultez le fichier `/proc/meminfo`. Comparez le contenu de ce fichier à la sortie de la commande `free` et identifiez quelle clé de `/proc/meminfo` correspond aux champs suivants dans la sortie de `free` :

Sortie de free	Champ de /proc/meminfo
total	
free	
shared	
buff/cache	
available	

## Résumé

Dans cette leçon, vous avez appris à connaître l'emplacement des programmes et de leurs fichiers de configuration dans un système Linux. Les faits importants à retenir sont les suivants :

- En principe, les programmes se trouvent dans une structure de répertoire à trois niveaux : `/`, `/usr` et `/usr/local`. Chacun de ces niveaux peut contenir des répertoires `bin` et `sbin`.
- Les fichiers de configuration sont stockés dans `/etc` et `~`.
- Les dotfiles sont des fichiers cachés qui commencent par un point (`.`).

Nous avons également discuté du noyau Linux. Les faits importants sont les suivants :

- Pour Linux, tout est un fichier.
- Le noyau Linux vit dans `/boot` avec d'autres fichiers liés au démarrage.
- Pour que les processus commencent à s'exécuter, le noyau doit d'abord être chargé dans une zone de mémoire protégée.
- La tâche du noyau est d'allouer les ressources du système aux processus.
- Le système de fichiers virtuel (ou pseudo) `/proc` stocke les données importantes du noyau et du système de manière volatile.

De même, nous avons exploré les périphériques matériels et appris ce qui suit :

- Le répertoire `/dev` stocke des fichiers spéciaux (alias nœuds) pour tous les périphériques matériels connectés : *périphériques à blocs* ou *périphériques à caractères*. Les premiers transfèrent les données par blocs ; les seconds, un caractère à la fois.
- Le répertoire `/dev` contient également d'autres fichiers spéciaux tels que `/dev/zero`, `/dev/null` ou `/dev/urandom`.
- Le répertoire `/sys` stocke des informations sur les dispositifs matériels classés par catégories.

Enfin, nous avons évoqué la mémoire. Nous avons appris :

- Qu'un programme s'exécute lorsqu'il est chargé en mémoire.
- Ce qu'est la RAM (Random Access Memory).
- Ce qu'est le Swap.
- Comment afficher l'utilisation de la mémoire.

Commandes utilisées dans cette leçon :

**cat**

Concatène/affiche le contenu du fichier.

**free**

Affiche la quantité de mémoire libre et utilisée dans le système.

**ls**

Liste le contenu du répertoire.

**which**

Affiche l'emplacement d'un programme.

## Réponses aux Exercices Guidés

1. Utilisez la commande `which` pour connaître l'emplacement des programmes suivants et compléter le tableau :

Programme	Commande <code>which</code>	Chemin vers l'exécutable (sortie)	L'utilisateur a besoin de privilèges de root ?
<code>swapon</code>	<code>which swapon</code>	<code>/sbin/swapon</code>	Oui
<code>kill</code>	<code>which kill</code>	<code>/bin/kill</code>	Non
<code>cut</code>	<code>which cut</code>	<code>/usr/bin/cut</code>	Non
<code>usermod</code>	<code>which usermod</code>	<code>/usr/sbin/usermod</code>	Oui
<code>cron</code>	<code>which cron</code>	<code>/usr/sbin/cron</code>	Oui
<code>ps</code>	<code>which ps</code>	<code>/bin/ps</code>	Non

2. Où se trouvent les fichiers suivants ?

Fichier	<code>/etc</code>	<code>~</code>
<code>.bashrc</code>	Non	Oui
<code>bash.bashrc</code>	Oui	Non
<code>passwd</code>	Oui	Non
<code>.profile</code>	Non	Oui
<code>resolv.conf</code>	Oui	Non
<code>sysctl.conf</code>	Oui	Non

3. Expliquez la signification des éléments numériques du fichier de noyau `vmlinux-4.15.0-50-generic` trouvé dans `/boot` :

Élément de numéro	Signification
4	Version du noyau
15	Révision majeure
0	Révision mineure
50	Numéro de correctif

4. Quelle commande utiliseriez-vous pour lister tous les disques durs et les partitions dans `/dev` ?

```
ls /dev/sd*
```



## Réponses aux Exercices d'Exploration

1. Les fichiers de périphériques pour les disques durs sont représentés sur la base des contrôleurs qu'ils utilisent comme nous l'avons vu `/dev/sd*` pour les lecteurs utilisant SCSI (Small Computer System Interface) et SATA (Serial Advanced Technology Attachment), mais

- Comment les anciens lecteurs IDE (Integrated Drive Electronics) étaient-ils représentés ?

`/dev/hd*`

- Et les lecteurs modernes NVMe (Non-Volatile Memory Express) ?

`/dev/nvme*`

2. Consultez le fichier `/proc/meminfo`. Comparez le contenu de ce fichier à la sortie de la commande `free` et identifiez quelle clé de `/proc/meminfo` correspond aux champs suivants dans la sortie de `free` :

Sortie de <code>free</code>	Champ de <code>/proc/meminfo</code>
total	MemTotal / SwapTotal
free	MemFree / SwapFree
shared	Shmem
buff/cache	Buffers, Cached et SReclaimable
available	MemAvailable



Linux  
Professional  
Institute

## 4.3 Leçon 2

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	4 Le Système d'Exploitation Linux
<b>Objectif :</b>	4.3 Lieux de Stockage des Données
<b>Leçon:</b>	2 sur 2

### Introduction

Après avoir exploré les programmes et leurs fichiers de configuration, nous apprendrons dans cette leçon comment les commandes sont exécutées en tant que processus. De même, nous commenterons la messagerie du système, l'utilisation de la mémoire tampon circulaire du noyau et de comment l'arrivée de `systemd` et de son démon de journalisation `journald` a changé la façon dont les choses ont été faites jusqu'à présent concernant la journalisation du système.

### Les Processus

Chaque fois qu'un utilisateur entre une commande, un programme est exécuté et un ou plusieurs processus sont générés.

Les processus existent dans une hiérarchie. Après le chargement du noyau en mémoire au démarrage, le premier processus est lancé qui a son tour démarre d'autres processus, qui, là encore, peuvent lancer d'autres processus. Chaque processus possède un identifiant unique (PID) et un identifiant de processus parent (PPID). Il s'agit d'entiers positifs qui sont attribués dans un ordre séquentiel.

## Exploration Dynamique des Processus : `top`

Vous pouvez obtenir une liste dynamique de tous les processus en cours avec la commande `top` :

```
$ top
```

```
top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14
Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,3 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache
KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
  436 carol    20   0   42696   3624   3060  R   0,7   0,4    0:00.30 top
     4 root      20   0      0      0      0  S   0,3   0,0    0:00.12 kworker/0:0
  399 root      20   0   95204   6748   5780  S   0,3   0,7    0:00.22 sshd
     1 root      20   0   56872   6596   5208  S   0,0   0,6    0:01.29 systemd
     2 root      20   0      0      0      0  S   0,0   0,0    0:00.00 kthreadd
     3 root      20   0      0      0      0  S   0,0   0,0    0:00.02 ksoftirqd/0
     5 root       0 -20      0      0      0  S   0,0   0,0    0:00.00 kworker/0:0H
     6 root      20   0      0      0      0  S   0,0   0,0    0:00.00 kworker/u2:0
     7 root      20   0      0      0      0  S   0,0   0,0    0:00.08 rcu_sched
     8 root      20   0      0      0      0  S   0,0   0,0    0:00.00 rcu_bh
     9 root      rt    0      0      0      0  S   0,0   0,0    0:00.00 migration/0
    10 root       0 -20      0      0      0  S   0,0   0,0    0:00.00 lru-add-drain
 (...)
```

Comme nous l'avons vu ci-dessus, `top` peut également nous donner des informations sur la consommation de mémoire et de CPU du système global ainsi que pour chaque processus.

`top` permet à l'utilisateur une certaine interaction.

Par défaut, la sortie est triée en fonction du pourcentage de temps CPU utilisé par chaque processus, par ordre décroissant. Ce comportement peut être modifié en appuyant sur les touches suivantes dans `top` :

**M**

Trie par utilisation de la mémoire.

**N**

Trie par numéro d'identification du processus.

**T**

Trie par durée d'exécution.

**P**

Trie par pourcentage d'utilisation du CPU.

Pour passer de l'ordre descendant à l'ordre ascendant, il suffit d'appuyer sur la touche **R**.

Une version plus conviviale de **top** est **htop**. Une autre alternative peut-être plus exhaustive est **atop**. Si elles ne sont pas déjà installées dans votre système, je vous encourage à utiliser votre gestionnaire de paquets pour les installer et les essayer.

## Capture Instantanée des Processus : **ps**

Une autre commande très utile pour obtenir des informations sur les processus est **ps**. Alors que **top** fournit des informations dynamiques, celles de **ps** sont statiques.

Si elle est invoquée sans options, la sortie de **ps** est assez discrète et ne concerne que les processus attachés au shell courant :

```
$ ps
  PID TTY          TIME CMD
 2318 pts/0    00:00:00 bash
 2443 pts/0    00:00:00 ps
```

Les informations affichées concernent l'identifiant du processus (PID), le terminal dans lequel le processus est exécuté (TTY), le temps CPU pris par le processus (TIME) et la commande qui a lancé le processus (CMD).

Une option utile pour **ps** est **-f** qui affiche la liste complète :

```
$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
carol        2318  1682  0 08:38 pts/1    00:00:00 bash
carol        2443  2318  0 08:46 pts/1    00:00:00 ps -f
```

En combinaison avec d'autres options, **-f** montre la relation entre les processus parents et enfants :

```
$ ps -uf
```

```
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
carol      2318  0.0  0.1  21336  5140 pts/1    Ss   08:38   0:00 bash
carol      2492  0.0  0.0  38304  3332 pts/1    R+   08:51   0:00 \_ ps -uf
carol      1780  0.0  0.1  21440  5412 pts/0    Ss   08:28   0:00 bash
carol      2291  0.0  0.7 305352 28736 pts/0    Sl+  08:35   0:00 \_ emacs index.en.adoc
-nw
(...)
```

De même, `ps` peut indiquer le pourcentage de mémoire utilisé lorsqu'il est invoqué avec l'option `-v` :

```
$ ps -v
```

```
PID TTY      STAT   TIME MAJFL   TRS   DRS   RSS %MEM COMMAND
1163 tty2    Ssl+   0:00      1    67 201224 5576  0.1 /usr/lib/gdm3/gdm-x-session (...)
(...)
```

#### NOTE

Une autre commande visuellement attrayante qui montre la hiérarchie des processus est `pstree`. Elle est livrée avec toutes les distributions majeures.

## Informations sur les Processus dans le Répertoire `/proc`

Nous avons déjà vu le système de fichiers `/proc`. `/proc` comprend un sous-répertoire numéroté pour chaque processus en cours d'exécution dans le système (le numéro est le `PID` du processus) :

```
carol@debian:~# ls /proc
```

```
1    108  13   17   21   27   354  41   665  8    9
10   109  14   173  22   28   355  42   7    804  915
103  11   140  18   23   29   356  428  749  810  918
104  111  148  181  24   3    367  432  75   811
105  112  149  19   244  349  370  433  768  83
106  115  15   195  25   350  371  5    797  838
107  12   16   2    26   353  404  507  798  899
(...)
```

Ainsi, toutes les informations relatives à un processus particulier sont incluses dans son répertoire. Lisons le contenu du premier processus dont le `PID` est 1 (la sortie a été tronquée pour des raisons de lisibilité) :

```
# ls /proc/1/
```

attr	cmdline	environ	io	mem	ns
autogroup	comm	exe	limits	mountinfo	numa_maps
auxv	coredump_filter	fd	loginuid	mounts	oom_adj
...					

Vous pouvez vérifier par exemple le processus exécutable :

```
# cat /proc/1/cmdline; echo
/sbin/init
```

Comme vous pouvez le voir, le binaire qui a lancé la hiérarchie des processus était `/sbin/init`.

#### NOTE

Les commandes peuvent être concaténées avec le point-virgule (;). Le but de l'utilisation de la commande `echo` ci-dessus est de fournir une nouvelle ligne. Essayez de lancer simplement `cat /proc/1/cmdline` pour voir la différence.

## La Charge du Système

Chaque processus d'un système peut potentiellement consommer des ressources du système. La valeur appelée "charge du système" tente d'agréger la charge globale du système en un seul indicateur numérique. Vous pouvez voir la charge actuelle avec la commande `uptime` :

```
$ uptime
22:12:54 up 13 days, 20:26, 1 user, load average: 2.91, 1.59, 0.39
```

Les trois derniers chiffres indiquent la moyenne de la charge du système pour la dernière minute (2,91), les cinq dernières minutes (1,59) et les quinze dernières minutes (0,39), respectivement.

Chacun de ces chiffres indique le nombre de processus en attente soit de ressources CPU, soit d'opérations d'entrée/sortie à réaliser. Cela signifie que ces processus étaient prêts à fonctionner s'ils avaient reçu les ressources respectives.

## Journalisation du Système et Messagerie du Système

Dès que le noyau et les processus commencent à s'exécuter et à communiquer entre eux, de nombreuses informations sont produites. La plupart d'entre elles sont enregistrées dans des fichiers appelés fichiers journaux (logs) ou, plus simplement, *journaux*.

Sans journalisation, la recherche d'un événement survenu sur un serveur donnerait aux administrateurs système de nombreux maux de tête, d'où l'importance de disposer d'un moyen

normalisé et centralisé de suivi des événements du système. En outre, les journaux sont déterminants et révélateurs en matière de dépannage et de sécurité, ainsi que des sources de données fiables pour comprendre les statistiques du système et faire des prévisions de tendances.

## Journalisation avec le Démon syslog

Traditionnellement, les messages du système sont gérés par l'utilitaire de journalisation standard syslog ou l'un de ses dérivés : syslog-ng ou rsyslog. Le démon de journalisation collecte les messages d'autres services et programmes et les stocke dans des fichiers de journalisation, généralement sous `/var/log`. Toutefois, certains services s'occupent de leurs propres journaux (exemple du serveur web Apache HTTPD). De même, le noyau Linux utilise un tampon circulaire pour stocker ses messages de journalisation.

### Fichiers Journaux dans `/var/log`

Comme les journaux sont des données qui varient dans le temps, ils se trouvent normalement dans `/var/log`.

Si vous explorez `/var/log`, vous vous rendrez compte que les noms des fichiers de journalisation sont à un certain degré auto explicatif. En voici quelques exemples :

#### `/var/log/auth.log`

Il stocke des informations sur l'authentification.

#### `/var/log/kern.log`

Il stocke les informations du noyau.

#### `/var/log/syslog`

Il stocke les informations du système.

#### `/var/log/messages`

Il stocke les données du système et des applications.

#### NOTE

Le nom exact et le contenu des fichiers journaux peuvent varier selon les distributions Linux.

## Accès aux Fichiers Journaux

Lorsque vous explorez les fichiers journaux, n'oubliez pas d'être root (si vous n'avez pas de droits de lecture) et d'utiliser un *pager* tel que `less` ;

**# less /var/log/messages**

```
Jun  4 18:22:48 debian liblogging-stdlog: [origin software="rsyslogd" swVersion="8.24.0" x-
pid="285" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Jun 29 16:57:10 debian kernel: [  0.000000] Linux version 4.9.0-8-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP
Debian 4.9.130-2 (2018-10-27)
Jun 29 16:57:10 debian kernel: [  0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-
8-amd64 root=/dev/sda1 ro quiet
```

Vous pouvez également utiliser la commande `tail` avec l'option `-f` pour lire les messages les plus récents du fichier et afficher dynamiquement les nouvelles lignes telles qu'elles sont ajoutées :

**# tail -f /var/log/messages**

```
Jul  9 18:39:37 debian kernel: [  2.350572] RAPL PMU: hw unit of domain psys 2^-0 Joules
Jul  9 18:39:37 debian kernel: [  2.512802] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input7
Jul  9 18:39:37 debian kernel: [  2.513861] Adding 1046524k swap on /dev/sda5. Priority:-
1 extents:1 across:1046524k FS
Jul  9 18:39:37 debian kernel: [  2.519301] hid-generic 0003:80EE:0021.0001:
input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
Jul  9 18:39:37 debian kernel: [  2.623947] snd_intel8x0 0000:00:05.0: white list rate for
1028:0177 is 48000
Jul  9 18:39:37 debian kernel: [  2.914805] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not
ready
Jul  9 18:39:39 debian kernel: [  4.937283] e1000: enp0s3 NIC Link is Up 1000 Mbps Full
Duplex, Flow Control: RX
Jul  9 18:39:39 debian kernel: [  4.938493] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link
becomes ready
Jul  9 18:39:40 debian kernel: [  5.315603] random: crng init done
Jul  9 18:39:40 debian kernel: [  5.315608] random: 7 urandom warning(s) missed due to
ratelimiting
```

Vous trouverez la sortie dans le format suivant :

- Horodatage
- Nom d'hôte d'où provient le message
- Nom du programme/service qui a généré le message
- Le PID du programme qui a généré le message
- Description de l'action qui a eu lieu



La plupart des fichiers journaux sont écrits en texte clair ; cependant, certains peuvent contenir des données binaires comme c'est le cas de `/var/log/wtmp` qui stocke les données relatives aux connexions réussies. Vous pouvez utiliser la commande `file` pour déterminer quel est le type de fichier :

```
$ file /var/log/wtmp
/var/log/wtmp: dBase III DBT, version number 0, next free block index 8
```

Ces fichiers sont normalement lus à l'aide de commandes spéciales. `last` est utilisée pour interpréter les données dans `/var/log/wtmp` :

```
$ last
carol    tty2          :0                Thu May 30 10:53    still logged in
reboot   system boot   4.9.0-9-amd64     Thu May 30 10:52    still running
carol    tty2          :0                Thu May 30 10:47 - crash (00:05)
reboot   system boot   4.9.0-9-amd64     Thu May 30 09:11    still running
carol    tty2          :0                Tue May 28 08:28 - 14:11 (05:42)
reboot   system boot   4.9.0-9-amd64     Tue May 28 08:27 - 14:11 (05:43)
carol    tty2          :0                Mon May 27 19:40 - 19:52 (00:11)
reboot   system boot   4.9.0-9-amd64     Mon May 27 19:38 - 19:52 (00:13)
carol    tty2          :0                Mon May 27 19:35 - down  (00:03)
reboot   system boot   4.9.0-9-amd64     Mon May 27 19:34 - 19:38 (00:04)
```

#### NOTE

Comme `/var/log/wtmp`, `/var/log/btmp` stocke les informations sur les tentatives de connexion échouées et la commande spéciale pour lire son contenu est `lastb`.

## La Rotation des Journaux

Les fichiers journaux peuvent se développer considérablement en quelques semaines ou mois et occuper tout l'espace disque disponible. Pour y remédier, on utilise l'utilitaire `logrotate`. Il met en œuvre la rotation ou le cycle des journaux, ce qui implique des actions telles que le déplacement des fichiers journaux vers un nouveau nom, leur archivage et/ou leur compression, parfois leur envoi par courrier électronique à l'administrateur système et finalement leur suppression au fur et à mesure qu'ils vieillissent. Les conventions utilisées pour nommer ces fichiers de log en rotation sont diverses (ajout d'un suffixe avec la date, par exemple) ; cependant, le simple ajout d'un suffixe avec un nombre entier est courant :

```
# ls /var/log/apache2/
access.log  error.log  error.log.1  error.log.2.gz  other_vhosts_access.log
```

Notez que le fichier `error.log.2.gz` a déjà été compressé avec `gzip` (d'où le suffixe `.gz`).

## Le Tampon Circulaire du Noyau (Kernel Ring Buffer)

Le tampon circulaire du noyau est une structure de données de taille fixe qui enregistre les messages de démarrage du noyau ainsi que tous les messages du noyau en direct. L'importante fonction de ce tampon est d'enregistrer tous les messages du noyau produits lors du démarrage lorsque `syslog` n'est pas encore disponible. La commande `dmesg` affiche le tampon circulaire du noyau (qui était également stocké dans `/var/log/dmesg`). En raison de l'extension du tampon circulaire, cette commande est normalement utilisée en combinaison avec l'utilitaire de filtrage de texte `grep` ou un *pager* tel que `less`. Par exemple, pour rechercher des messages de démarrage :

```
$ dmesg | grep boot
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.000000] smpboot: Allowing 1 CPUs, 0 hotplug CPUs
[ 0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.144986] AppArmor: AppArmor disabled by boot time parameter
(...)
```

### NOTE

Au fur et à mesure que le tampon circulaire du noyau se développe avec de nouveaux messages, les plus anciens s'effacent.

## Le Journal du Système : systemd-journald

À partir de 2015, `systemd` a remplacé `SysV Init` comme gestionnaire de système et de services *de facto* dans la plupart des grandes distributions Linux. En conséquence, le démon de journalisation `journald` est devenu le composant de journalisation standard, remplaçant `syslog` dans la plupart des aspects. Les données ne sont plus stockées en texte brut mais sous forme binaire. Ainsi, l'utilitaire `journalctl` est nécessaire pour lire les journaux. En outre, `journald` est compatible avec `syslog` et peut être intégré à `syslog`.

`journalctl` est l'utilitaire pour lire et interroger la base de données des journaux de `Systemd`. S'il est invoqué sans options, il affiche le journal dans son intégralité :

```
# journalctl
-- Logs begin at Tue 2019-06-04 17:49:40 CEST, end at Tue 2019-06-04 18:13:10 CEST. --
jun 04 17:49:40 debian systemd-journald[339]: Runtime journal (/run/log/journal/) is 8.0M,
max 159.6M, 151.6M free.
jun 04 17:49:40 debian kernel: microcode: microcode updated early to revision 0xcc, date =
```

```
2019-04-01
```

```
Jun 04 17:49:40 debian kernel: Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org)
(gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
```

```
Jun 04 17:49:40 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64
```

```
root=/dev/sda1 ro quiet
```

```
(...)
```

Cependant, s'il est invoqué avec les options `-k` ou `--dmesg`, il sera équivalent à l'utilisation de la commande `dmesg` :

```
# journalctl -k
```

```
[ 0.000000] Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version
6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13)
```

```
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-
441f-b8f5-8061c0034c74 ro quiet
```

```
(...)
```

D'autres options intéressantes pour `journalctl` sont :

### **-b, --boot**

Elle affiche les informations de démarrage.

### **-u**

Elle affiche des messages concernant une unité spécifique. En gros, une unité peut être définie comme toute ressource gérée par `systemd`. Par exemple, `journalctl -u apache2.service` est utilisé pour lire les messages concernant le serveur web `apache2`.

### **-f**

Elle affiche les messages les plus récents du journal et continue d'afficher les nouvelles entrées telles qu'elles sont ajoutées au journal, comme le ferait `tail -f`.

## Exercices Guidés

1. Consultez la sortie suivante de `top` et répondez aux questions suivantes :

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running, 71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 1020332 total,  698700 free,  170664 used,  150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,      0 used.  710956 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424 34256 S   6.3  13.0   1:47.24 ntopng
  444 www-data   20   0  364780   4132   2572 S   0.3   0.4   0:00.44 apache2
  734 root       20   0   95212   7004   6036 S   0.3   0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R   0.3   0.4   0:00.03 top
    1 root      20   0   56988   6688   5240 S   0.0   0.7   0:00.42 systemd
    2 root      20   0        0        0        0 S   0.0   0.0   0:00.00 kthreadd
    3 root      20   0        0        0        0 S   0.0   0.0   0:00.09 ksoftirqd/0
    4 root      20   0        0        0        0 S   0.0   0.0   0:00.87 kworker/0:0
(...)
```

- Quels processus ont été lancés par l'utilisateur `carol` ?
- Quel répertoire virtuel de `/proc` devez-vous visiter pour rechercher des données concernant la commande `top` ?
- Quel processus a été lancé en premier ? Comment pouvez-vous le savoir ?
- Complétez le tableau en précisant dans quelle zone de la sortie de `top` se trouvent les informations suivantes :

Informations sur ...	Zone du résumé	Zone d'activité
Mémoire		
Swap		
PID		

Informations sur ...	Zone du résumé	Zone d'activité
Temps CPU		
Commandes		

2. Quelle commande est utilisée pour lire les journaux binaires suivants ?

- `/var/log/wtmp`

- `/var/log/btmp`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

3. En combinaison avec `grep`, quelles commandes utiliseriez-vous pour trouver les informations suivantes sur votre système Linux ?

- Quand le système a été redémarré pour la dernière fois (`wtmp`)

- Quels sont les disques durs installés (`kern.log`)

- Date de la dernière connexion (`auth.log`)

4. Quelles sont les deux commandes que vous utiliseriez pour afficher le tampon circulaire du noyau ?

5. Indiquez où se trouvent les messages de journal suivant :

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

<code>/var/log/auth.log</code>	
<code>/var/log/kern.log</code>	
<code>/var/log/syslog</code>	
<code>/var/log/messages</code>	

- Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 14:02:53 debian sudo: pam\_unix(sudo:session): session opened for user root by carol(uid=0)

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

6. Exécutez `journalctl` pour obtenir des informations sur les unités suivantes ?

Unité	Commande
ssh	
networking	
rsyslog	
cron	

## Exercices d'Exploration

1. Reconsidérez la sortie du `top` des exercices guidés et répondez aux questions suivantes :

- Quelles sont les deux étapes à suivre pour tuer le serveur web *apache* ?

- Dans la zone de résumé, comment pourriez-vous afficher les informations sur la mémoire physique et Swap à l'aide de barres de progression ?

- Maintenant, trie les processus en fonction de l'utilisation de la mémoire :

- Maintenant que vous avez des informations sur la mémoire affichée dans les barres de progression et des processus triés par utilisation de la mémoire, enregistrez ces configurations afin de les obtenir par défaut la prochaine fois que vous utiliserez `top` :

- Quel fichier stocke les paramètres de configuration de `top` ? Où se trouve-t-il ? Comment pouvez-vous vérifier son existence ?

2. Apprenez la commande `exec` dans Bash. Essayez de démontrer sa fonctionnalité en démarrant une session Bash, en trouvant le processus Bash avec `ps`, puis exécutez `exec /bin/sh` et recherchez à nouveau le processus avec le même PID.

3. Suivez ces étapes pour explorer les événements du noyau et la gestion dynamique des périphériques par `udev` :

- Branchez une clé USB sur votre ordinateur. Lancez `dmesg` et faites attention aux dernières lignes. Quelle est la ligne la plus récente ?

- En gardant à l'esprit la sortie de la commande précédente, exécutez `ls /dev/sd*` et assurez-vous que votre clé USB apparaît dans la liste. Quelle est la sortie ?

- Maintenant, retirez la clé USB et lancez à nouveau `dmesg`. Comment est la ligne la plus récente ?

- Exécutez à nouveau `ls /dev/sd*` et assurez-vous que votre appareil a disparu de la liste. Quel est le résultat ?



## Résumé

Dans le contexte du stockage des données, les sujets suivants ont été abordés dans cette leçon : la gestion des processus, la journalisation et la messagerie du système.

En ce qui concerne la gestion des processus, nous avons appris ce qui suit :

- Les programmes génèrent des processus et les processus existent dans une hiérarchie.
- Chaque processus possède un identifiant unique (PID) et un identifiant de processus parent (PPID).
- `top` est une commande très utile pour explorer de manière dynamique et interactive les processus en cours du système.
- `ps` peut être utilisé pour obtenir une capture instantanée des processus en cours d'exécution dans le système.
- Le répertoire `/proc` comprend des répertoires pour chaque processus en cours d'exécution dans le système, nommés d'après leur PID.
- Le concept de charge moyenne du système est très utile pour vérifier l'utilisation ou la surcharge du processeur.

En ce qui concerne la journalisation système, nous devons nous souvenir :

- Qu'un journal est un fichier dans lequel sont enregistrés les événements du système. Les journaux sont d'une valeur inestimable pour le dépannage.
- Que la journalisation a traditionnellement été assurée par des services spéciaux tels que `syslog`, `syslog-ng` ou `rsyslog`. Néanmoins, certains programmes utilisent leurs propres démons de journalisation.
- Que comme les journaux sont des données variables, ils sont conservés dans `/var` et parfois leurs noms peuvent vous donner un indice sur leur contenu (`kern.log`, `auth.log`, etc.)
- Que la plupart des journaux sont écrits en texte simple et peuvent être lus avec n'importe quel éditeur de texte, à condition que vous disposiez des autorisations nécessaires. Toutefois, certains sont binaires et doivent être lus à l'aide de commandes spéciales.
- Que pour éviter les problèmes d'espace disque, la *rotation des journaux* est effectuée par l'utilitaire `logrotate`.
- Quant au noyau, il utilise une structure de données circulaire, le tampon circulaire, où les messages de démarrage sont conservés (les anciens messages s'effacent avec le temps).
- Que le gestionnaire du système et des services `systemd` a remplacé le `System V init` dans

pratiquement toutes les distributions, journald devenant le service de journalisation standard.

- Que pour lire le journal de Systemd, l'utilitaire `journalctl` est nécessaire.

Commandes utilisées dans cette leçon :

### **cat**

Concatène/affiche le contenu du fichier.

### **dmesg**

Affiche le tampon circulaire du noyau.

### **echo**

Affiche une ligne de texte ou une nouvelle ligne.

### **file**

Détermine le type de fichier.

### **grep**

Affiche des lignes correspondant à un motif.

### **last**

Affiche une liste des derniers utilisateurs connectés.

### **less**

Affiche le contenu du fichier une page à la fois.

### **ls**

Liste le contenu du répertoire.

### **journalctl**

Interroge le journal de `systemd`.

### **tail**

Affiche les dernières lignes d'un fichier.

# Réponses aux Exercices Guidés

1. Consultez la sortie suivante de `top` et répondez aux questions suivantes :

```
carol@debian:~$ top
```

```
top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,   0.0 ni, 98.6 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
KiB Mem : 1020332 total,  698700 free,  170664 used,  150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,      0 used.  710956 avail Mem
```

```

PID USER      PR  NI   VIRT   RES    SHR S  %CPU  %MEM    TIME+  COMMAND
605 nobody    20   0 1137620 132424 34256 S   6.3  13.0   1:47.24 ntopng
444 www-data   20   0  364780   4132  2572 S   0.3   0.4   0:00.44 apache2
734 root       20   0   95212   7004  6036 S   0.3   0.7   0:00.36 sshd
887 carol     20   0   46608   3680  3104 R   0.3   0.4   0:00.03 top
   1 root      20   0   56988   6688  5240 S   0.0   0.7   0:00.42 systemd
   2 root      20   0        0     0     0 S   0.0   0.0   0:00.00 kthreadd
   3 root      20   0        0     0     0 S   0.0   0.0   0:00.09 ksoftirqd/0
   4 root      20   0        0     0     0 S   0.0   0.0   0:00.87 kworker/0:0
(...)

```

- Quels processus ont été lancés par l'utilisateur `carol`?

Réponse : Un seul : `top`.

- Quel répertoire virtuel de `/proc` devez-vous visiter pour rechercher des données concernant la commande `top` ?

Réponse : `/proc/887`

- Quel processus a été lancé en premier ? Comment pouvez-vous le savoir ?

Réponse : `systemd`. Parce que c'est celui qui a le PID n°1.

- Complétez le tableau en précisant dans quelle zone de la sortie de `top` se trouvent les informations suivantes :

Informations sur ...	Zone de résumé	Zone d'activité
Mémoire	Oui	Oui

Informations sur ...	Zone de résumé	Zone d'activité
Swap	Oui	Non
PID	Non	Oui
Temps CPU	Oui	Oui
Commandes	Non	Oui

2. Quelle commande est utilisée pour lire les journaux binaires suivants ?

- `/var/log/wtmp`

Réponse : `last`

- `/var/log/btmp`

Réponse : `lastb`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

Réponse : `journalctl`

3. En combinaison avec `grep`, quelles commandes utiliseriez-vous pour trouver les informations suivantes sur votre système Linux ?

- Quand le système a été redémarré pour la dernière fois (`wtmp`)

Réponse : `last`

- Quels sont les disques durs installés (`kern.log`)

Réponse : `less /var/log/kern.log`

- La date de la dernière connexion (`auth.log`)

Réponse : `less /var/log/auth.log`

4. Quelles sont les deux commandes que vous utiliseriez pour afficher le tampon circulaire du noyau ?

`dmesg` et `journalctl -k` (également `journalctl --dmesg`).

5. Indiquez où se trouvent les messages de journal suivants :

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	X
/var/log/messages	

- Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver

/var/log/auth.log	
/var/log/kern.log	X
/var/log/syslog	
/var/log/messages	X

Jul 10 14:02:53 debian sudo: pam\_unix(sudo:session): session opened for user root by carol(uid=0)

/var/log/auth.log	X
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	X

6. Exécutez `journalctl` pour obtenir des informations sur les unités suivantes :

Unité	Commande
ssh	<code>journalctl -u ssh.service</code>
networking	<code>journalctl -u networking.service</code>
rsyslog	<code>journalctl -u rsyslog.service</code>

Unité	Commande
cron	journalctl -u cron.service

# Réponses aux Exercices d'Exploration

1. Reconsidérez la sortie du `top` des exercices guidés et répondez aux questions suivantes :

- Quelles sont les deux étapes à suivre pour tuer le serveur web *apache* ?

D'abord, appuyez sur `k` ; puis fournissez une valeur `kill`.

- Dans la zone de résumé, comment pourriez-vous afficher les informations sur la mémoire physique et le Swap à l'aide de barres de progression ?

En appuyant sur `m` une ou deux fois.

- Maintenant, triez les processus en fonction de l'utilisation de la mémoire :

`M`

- Maintenant que vous avez des informations sur la mémoire affichée dans les barres de progression et des processus triés par utilisation de la mémoire, enregistrez ces configurations afin de les obtenir par défaut la prochaine fois que vous utiliserez `top` :

`W`

- Quel fichier stocke les paramètres de configuration de `top` ? Où se trouve-t-il ? Comment pouvez-vous vérifier son existence ?

Le fichier est `~/.config/procps/toprc` et se trouve dans le répertoire personnel de l'utilisateur (`~`). Comme il s'agit d'un fichier caché (il réside dans un répertoire dont le nom commence par un point), nous pouvons vérifier son existence avec `ls -a` (listage de tous les fichiers). Ce fichier peut être généré en appuyant sur `Shift + W` dans `top`.

2. Apprenez la commande `exec` dans Bash. Essayez de démontrer sa fonctionnalité en démarrant une session Bash, en trouvant le processus Bash avec `ps`, puis exécutez `exec /bin/sh` et recherchez à nouveau le processus avec le même PID.

`exec` remplace un processus par une autre commande. Dans l'exemple suivant, nous pouvons voir que le processus Bash est remplacé par `/bin/sh` (au lieu que `/bin/sh` devienne un processus enfant) :

```
$ echo $$
19877
$ ps auxf | grep 19877 | head -1
carol 19877 0.0 0.0 7448 3984 pts/25 Ss 21:17 0:00 \_ bash
```

```
$ exec /bin/sh
sh-5.0$ ps auxf | grep 19877 | head -1
carol 19877 0.0 0.0 7448 3896 pts/25 Ss 21:17 0:00 \_ /bin/sh
```

3. Suivez ces étapes pour explorer les événements du noyau et la gestion dynamique des périphériques par udev :

- Branchez une clé USB à votre ordinateur. Lancez `dmesg` et faites attention aux dernières lignes. Quelle est la ligne la plus récente ?

Vous devriez obtenir quelque chose du genre `[1967.700468] sd 6:0:0:0 : [sdb] Attached SCSI removable disk`.

- En gardant à l'esprit la sortie de la commande précédente, exécutez `ls /dev/sd*` et assurez-vous que votre clé USB apparaît dans la liste. Quelle est la sortie ?

Selon le nombre de périphériques connectés à votre système, vous devriez obtenir quelque chose comme `/dev/sda /dev/sda1 /dev/sdb /dev/sdb1 /dev/sdb2`. Dans notre cas, nous trouvons notre clé USB (`/dev/sdb`) et ses deux partitions (`/dev/sdb1` et `/dev/sdb2`).

- Maintenant, retirez la clé USB et lancez à nouveau `dmesg`. Comment est la ligne la plus récente ?

Vous devriez obtenir quelque chose du genre `[2458.881695] usb 1-9 : USB disconnect, dispositif numéro 6`.

- Exécutez à nouveau `ls /dev/sd*` et assurez-vous que votre appareil a disparu de la liste. Quel est le résultat ?

Dans notre cas, `/dev/sda /dev/sda1`.





## 4.4 Intégration au réseau

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 4.4

### Valeur

2

### Domaines de connaissance les plus importants

- Internet, réseau, routeurs
- Interrogation de la configuration DNS cliente
- Interrogation de la configuration du réseau

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `route`, `ip route show`
- `ifconfig`, `ip addr show`
- `netstat`, `ss`
- `/etc/resolv.conf`, `/etc/hosts`
- IPv4, IPv6
- `ping`
- `host`



Linux  
Professional  
Institute

## 4.4 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	4 Le Système d'Exploitation Linux
<b>Objectif :</b>	4.4 Votre Ordinateur dans le Réseau
<b>Leçon :</b>	1 sur 1

### Introduction

Dans le monde d'aujourd'hui, les appareils informatiques de toute sorte échangent des informations par le biais de réseaux. Au cœur même du concept de réseau informatique se trouvent les connexions physiques entre un appareil et son ou ses pairs. Ces connexions sont appelées des *liens*, et elles constituent la connexion la plus élémentaire entre deux appareils différents. Les liens peuvent être établis par le biais de divers supports, tels que les câbles de cuivre, les fibres optiques, les ondes radio ou les lasers.

Chaque lien est relié à une interface d'un appareil. Chaque appareil peut avoir plusieurs interfaces et donc être connecté à plusieurs liens. Grâce à ces liens, les ordinateurs peuvent former un réseau ; une petite communauté d'appareils qui peuvent se connecter directement les uns aux autres. Il existe de nombreux exemples de tels réseaux dans le monde. Pour pouvoir communiquer au-delà de la portée d'un réseau à couche de liaison, les appareils utilisent des routeurs. Pensez aux réseaux de la couche de liaison comme des îlots reliés par des routeurs qui relient les îles comme des ponts par lesquels les informations doivent voyager pour atteindre un appareil qui fait partie d'une île éloignée.

Ce modèle conduit à plusieurs couches différentes de mise en réseau :

## Couche de liaison

Gère la communication entre les appareils directement connectés.

## Couche réseau

Gère le routage en dehors des réseaux individuels et l'adressage unique des appareils au-delà d'un réseau à couche de liaison unique.

## Couche application

Permet aux programmes individuels de se connecter les uns aux autres.

Lorsqu'ils ont été inventés, les réseaux informatiques utilisaient les mêmes méthodes de communication que les téléphones en ce sens qu'ils étaient à commutation de circuits. Cela signifie qu'un lien direct et dédié devait être établi entre deux nœuds pour qu'ils puissent communiquer. Cette méthode fonctionnait bien, mais elle nécessitait tout l'espace d'un lien donné pour que seuls deux hôtes puissent communiquer.

Finalement, les réseaux informatiques sont passés à la *commutation par paquets*. Dans cette méthode, les données sont regroupées avec un en-tête, qui contient des informations sur la provenance et la destination des informations. Les informations sur le contenu réel sont contenues dans ce cadre et envoyées par le biais du lien vers le destinataire indiqué dans l'en-tête du cadre. Cela permet à plusieurs appareils de partager un même lien et de communiquer presque simultanément.

## Réseau de Couche de Liaison

Le rôle de tout paquet est de transporter l'information de la source à sa destination par un lien reliant les deux appareils. Ces appareils ont besoin d'un moyen de s'identifier l'un à l'autre. C'est le but d'une *adresse de couche de liaison*. Dans un réseau ethernet, les *adresses MAC* (Media Access Control Addresses) sont utilisées pour identifier les dispositifs individuels. Une adresse MAC se compose de 48 bits. Elles ne sont pas nécessairement uniques au niveau mondial et ne peuvent pas être utilisées pour s'adresser à des pairs en dehors de la liaison en cours. Ainsi, ces adresses ne peuvent pas être utilisées pour acheminer des paquets vers d'autres liens. Le destinataire d'un paquet vérifie si l'adresse de destination correspond à sa propre couche de liaison et, si c'est le cas, il traite le paquet. Dans le cas contraire, le paquet est abandonné. L'exception à cette règle concerne les *paquets de diffusion* (un paquet envoyé à tout le monde dans un réseau local donné) qui sont toujours acceptés.

La commande `ip link show` affiche une liste de toutes les interfaces réseau disponibles et leurs adresses de couche de liaison ainsi que d'autres informations sur la taille maximale des paquets :

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
```

La sortie ci-dessus montre que l'appareil a deux interfaces, `lo` et `ens33`. `lo` est l'*interface de rebouclage* et a l'adresse MAC `00:00:00:00:00:00` alors que `ens33` est une interface ethernet et a l'adresse MAC `00:0c:29:33:3b:25`.

## Réseau IPv4

Pour visiter des sites web tels que Google ou Twitter, pour vérifier les courriels ou pour permettre aux entreprises de se connecter entre elles, les paquets doivent pouvoir passer d'un réseau de couche de liaison à un autre. Souvent, ces réseaux ne sont connectés qu'indirectement, avec plusieurs réseaux intermédiaires de la couche de liaison que les paquets doivent traverser pour atteindre leur destination réelle.

Les adresses de la couche de liaison d'une interface réseau ne peuvent pas être utilisées en dehors de ce réseau spécifique de la couche de liaison. Comme cette adresse n'a pas d'importance pour les appareils des autres réseaux de la couche de liaison, il faut une forme d'adresses uniques au niveau mondial pour mettre en œuvre le routage. Ce système d'adressage, ainsi que le concept général de routage, est mis en œuvre par le *protocole Internet* (IP).

### NOTE

Un *protocole* est un ensemble de procédures permettant de faire quelque chose de manière que toutes les parties qui le suivent soient compatibles entre elles. Un protocole peut être considéré comme la définition d'une norme. En informatique, le protocole Internet est une norme acceptée par tous, de sorte que différents appareils produits par différents fabricants peuvent tous communiquer entre eux.

## Adresses IPv4

Les adresses IP, comme les adresses MAC, sont un moyen d'indiquer d'où vient un paquet de données et où il va. IPv4 était le protocole d'origine. Les adresses IPv4 ont une largeur de 32 bits, ce qui donne un nombre maximum théorique de 4 294 967 296 adresses. Toutefois, le nombre de ces adresses utilisables par les appareils est beaucoup plus faible, car certaines plages sont réservées à des cas d'utilisation particuliers tels que les adresses de diffusion (qui sont utilisées pour atteindre tous les participants d'un réseau spécifique), les adresses de multidiffusion

(semblables aux adresses de diffusion, mais un appareil doit se connecter comme une radio) ou celles réservées à un usage privé.

Dans leur format lisible par l'homme, les adresses IPv4 sont désignées par quatre chiffres séparés par un point. Chaque chiffre peut aller de 0 à 255. Par exemple, prenons l'adresse IP suivante :

```
192.168.0.1
```

Techniquement, chacun de ces chiffres représente huit bits individuels. Ainsi, cette adresse pourrait également être écrite comme ceci :

```
11000000.10101000.00000000.00000001
```

Dans la pratique, on utilise la notation décimale telle que vue ci-dessus. Cependant, la représentation par bits reste importante pour comprendre le découpage en sous-réseaux (*sous-réseautage*).

## Sous-réseaux IPv4

Afin de permettre le routage, les adresses IP peuvent être divisées en deux parties : la partie réseau et la partie hôte. La partie réseau identifie le réseau sur lequel l'appareil se trouve et est utilisée pour acheminer les paquets vers ce réseau. La partie hôte est utilisée pour identifier spécifiquement un appareil donné sur un réseau et pour remettre le paquet à son destinataire spécifique une fois qu'il a atteint son réseau de couche de liaison.

Les adresses IP peuvent être divisées en sous-réseaux et en hôtes à tout moment. Le *masque de sous-réseau* définit l'endroit où cette division se produit. Revenons sur la représentation binaire de l'adresse IP de l'exemple précédent :

```
11000000.10101000.00000000.00000001
```

Maintenant, pour cette adresse IP, le masque de sous-réseau met à 1 chaque bit appartenant à la partie réseau et à 0 chaque bit appartenant à la partie hôte :

```
11111111.11111111.11111111.00000000
```

Dans la pratique, le masque de sous-réseau s'écrit en notation décimale :

```
255.255.255.0
```

Cela signifie que ce réseau s'étend de 192.168.0.0 à 192.168.0.255. Notez que les trois premiers chiffres, dont tous les bits sont définis dans le masque de sous-réseau, restent inchangés dans les adresses IP.

Enfin, il existe une autre notation pour le masque de sous-réseau, appelée *Classless Inter-Domain Routing* (CIDR). Cette notation indique simplement combien de bits sont définis dans le masque de sous-réseau et ajoute ce nombre à l'adresse IP. Dans l'exemple, 24 des 32 bits sont définis à 1 dans le masque de sous-réseau. Cela peut être exprimé en notation CIDR sous la forme 192.168.0.1/24

## Adresses IPv4 Privées

Comme mentionné précédemment, certaines sections de l'espace d'adressage IPv4 sont réservées à des cas d'utilisation particuliers. L'un de ces cas d'utilisation est l'attribution d'adresses privées. Les sous-réseaux suivants sont réservés à l'adressage privé :

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Les adresses de ces sous-réseaux peuvent être utilisées par tout le monde. Toutefois, ces sous-réseaux ne peuvent pas être acheminés sur l'internet public car ils sont potentiellement utilisés par de nombreux réseaux en même temps.

Aujourd'hui, la plupart des réseaux utilisent ces adresses internes. Elles permettent la communication interne sans qu'il soit nécessaire d'attribuer une adresse externe. Aujourd'hui, la plupart des connexions Internet sont simplement fournies avec une seule adresse IPv4 externe. Les routeurs font correspondre toutes les adresses internes à cette adresse IP externe unique lorsqu'ils transfèrent des paquets vers l'internet. C'est ce qu'on appelle la *traduction d'adresse réseau* (NAT : Network Address Translation). Le cas particulier de NAT, où un routeur mappe les adresses internes à une seule adresse IP externe, est parfois appelé *masquage*. Cela permet à n'importe quel appareil du réseau interne d'établir de nouvelles connexions avec n'importe quelle adresse IP globale sur l'internet.

### NOTE

Avec le masquage, les appareils internes ne peuvent pas être référencés sur Internet car ils n'ont pas d'adresse globalement valable. Toutefois, il ne s'agit pas d'un dispositif de sécurité. Même en cas de masquage, un pare-feu est toujours nécessaire.

## Configuration de l'Adresse IPv4

Il y a deux façons principales de configurer les adresses IPv4 sur un ordinateur. Soit en attribuant les adresses manuellement, soit en utilisant le protocole DHCP (*Dynamic Host Configuration Protocol*) pour une configuration automatique.

Lorsqu'on utilise le DHCP, un serveur central contrôle quelles adresses sont distribuées à quels appareils. Le serveur peut également fournir aux appareils d'autres informations sur le réseau, telles que les adresses IP des serveurs DNS, l'adresse IP du routeur par défaut ou, dans le cas de configurations plus complexes, pour démarrer un système d'exploitation à partir du réseau. Le DHCP est activé par défaut sur de nombreux systèmes, vous aurez donc probablement déjà une adresse IP lorsque vous serez connecté à un réseau.

Les adresses IP peuvent également être ajoutées manuellement à une interface à l'aide de la commande `ip addr add`. Ici, nous ajoutons l'adresse `192.168.0.5` à l'interface `ens33`. Le réseau utilise le masque de sous-réseau `255.255.255.0` qui est égal à `/24` en notation CIDR :

```
$ sudo ip addr add 192.168.0.5/255.255.255.0 dev ens33
```

Nous pouvons maintenant vérifier que l'adresse a été ajoutée en utilisant la commande `ip addr show` :

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

La sortie ci-dessus montre à la fois la commande ainsi que les interfaces `lo` et `ens33` avec l'adresse assignée.

Pour vérifier l'accessibilité d'un appareil, la commande `ping` peut être utilisée. Elle envoie un type

spécial de message appelé *echo request* dans lequel l'expéditeur demande une réponse au destinataire. Si la connexion entre les deux appareils peut être établie avec succès, le destinataire enverra un *echo reply*, vérifiant ainsi la connexion :

```
$ ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.16 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.85 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=3.41 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 1.849/2.473/3.410/0.674 ms
```

Le paramètre `-c 3` fait que le `ping` s'arrête après l'envoi de trois *echo requests*. Sinon, le `ping` continue à s'exécuter indéfiniment et doit être arrêté en appuyant sur `Ctrl + C`.

## Routage IPv4

Le routage est le processus par lequel un paquet passe du réseau source au réseau de destination. Chaque appareil gère une table de routage qui contient des informations sur le réseau IP qui peut être directement atteint par le biais de la connexion de l'appareil aux réseaux de la couche de liaison et sur le réseau IP qui peut être atteint en transmettant les paquets à un routeur. Enfin, une *route par défaut* définit un routeur qui reçoit tous les paquets qui ne correspondent à aucune autre route.

Lorsqu'il établit une connexion, l'appareil recherche l'adresse IP de la cible dans sa table de routage. Si une entrée correspond à l'adresse, le paquet est soit envoyé au réseau de la couche de liaison correspondante, soit transmis au routeur indiqué dans la table de routage.

Les routeurs eux-mêmes maintiennent également des tables de routage. Lorsqu'il reçoit un paquet, un routeur recherche également l'adresse de destination dans sa propre table de routage et envoie le paquet au routeur suivant. Cette opération est répétée jusqu'à ce que le paquet arrive au routeur sur le réseau de destination. Chaque routeur impliqué dans ce voyage est appelé un *hop*. Ce dernier routeur trouve une liaison directe pour l'adresse cible dans sa table de routage et envoie les paquets à son interface cible.

La plupart des réseaux domestiques n'ont qu'une seule issue, le routeur singulier qui provient du *fournisseur d'accès à Internet* (FAI). Dans ce cas, un appareil se contente de transmettre tous les paquets qui ne sont pas destinés au réseau interne directement au routeur domestique, qui les envoie ensuite au routeur du fournisseur pour qu'ils soient retransmis. Il s'agit d'un exemple de



route par défaut.

La commande `ip route show` liste la table de routage IPv4 actuelle :

```
$ ip route show
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Pour ajouter une route par défaut, il suffit d'indiquer l'adresse interne du routeur qui sera la passerelle par défaut. Si, par exemple, le routeur a l'adresse `192.168.0.1`, alors la commande suivante le configure comme route par défaut :

```
$ sudo ip route add default via 192.168.0.1
```

Pour vérifier, lancez à nouveau `ip route show` :

```
$ ip route show
default via 192.168.0.1 dev ens33
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

## Réseau IPv6

L'IPv6 a été conçu pour pallier les lacunes de l'IPv4, principalement le manque d'adresses à mesure que de plus en plus d'appareils étaient mis en ligne. Toutefois, IPv6 comprend également d'autres caractéristiques telles que de nouveaux protocoles pour la configuration automatique du réseau. Au lieu de 32 bits par adresse, IPv6 en utilise 128. Cela permet d'obtenir environ  $2^{128}$  adresses. Toutefois, comme pour l'IPv4, le nombre d'adresses utilisables uniques au niveau mondial est beaucoup plus faible, car certaines parties de l'attribution sont réservées à d'autres usages. Ce grand nombre d'adresses signifie qu'il y a plus qu'assez d'adresses publiques pour chaque appareil actuellement connecté à l'internet et pour beaucoup d'autres à venir, ce qui réduit la nécessité de masquer les adresses et ses problèmes tels que le retard de traduction et l'impossibilité de se connecter directement à des appareils masqués.

## Adresses IPv6

Les adresses utilisent 8 groupes de 4 chiffres hexadécimaux séparés par deux points :

```
2001:0db8:0000:abcd:0000:0000:0000:7334
```

**NOTE**

Les chiffres hexadécimaux vont de 0 à f, de sorte que chaque chiffre peut contenir l'une des 16 valeurs différentes.

Pour faciliter la tâche, les zéros de tête de chaque groupe peuvent être supprimés lorsqu'ils sont écrits, mais chaque groupe doit contenir au moins un chiffre :

```
2001:db8:0:abcd:0:0:0:7334
```

Lorsque plusieurs groupes ne contenant que des zéros se suivent directement, ils peuvent être entièrement remplacés par :: :

```
2001:db8:0:abcd::7334
```

Toutefois, cela ne peut se produire qu'une seule fois par adresse.

## Préfixe IPv6

Les 64 premiers bits d'une adresse IPv6 sont connus comme le *préfixe de routage*. Le préfixe est utilisé par les routeurs pour déterminer à quel réseau appartient un appareil et donc sur quel chemin les données doivent être envoyées. Le sous-réseautage se produit toujours à l'intérieur du préfixe. Les FAI distribuent généralement les préfixes /48 ou /58 à leurs clients, leur laissant 16 ou 8 bits pour leur sous-réseautage interne.

Il existe trois grands types de préfixes dans IPv6 :

### Adresse Mondialement Unique

Dans laquelle le préfixe est attribué à partir des blocs réservés aux adresses globales. Ces adresses sont valables sur l'ensemble de l'internet.

### Adresse Unique Localement

Ne peut pas être acheminée sur Internet. Elles peuvent cependant être acheminées en interne au sein d'une organisation. Ces adresses sont utilisées au sein d'un réseau pour garantir que les appareils ont toujours une adresse même lorsqu'il n'y a pas de connexion internet. Elles sont l'équivalent des plages d'adresses privées à partir d'IPv4. Les 8 premiers bits sont toujours fc ou fd, suivis de 40 bits générés de façon aléatoire.

## Adresse Locale du Lien

Ne sont valables que sur un lien particulier. Chaque interface réseau compatible IPv6 possède une telle adresse, qui commence par `fe80`. Ces adresses sont utilisées en interne par IPv6 pour demander des adresses supplémentaires par configuration automatique et pour trouver d'autres ordinateurs sur le réseau à l'aide du protocole Neighbor Discovery.

## Identificateur d'Interface IPv6

Alors que le préfixe détermine dans quel réseau un appareil réside, l'identifiant d'interface est utilisé pour énumérer les appareils d'un réseau. Les 64 derniers bits d'une adresse IPv6 forment l'identificateur d'interface, tout comme les derniers bits d'une adresse IPv4.

Lorsqu'une adresse IPv6 est attribuée manuellement, l'identifiant de l'interface est défini comme faisant partie de l'adresse. Lorsque la configuration automatique de l'adresse est utilisée, l'identificateur de l'interface est soit choisi au hasard, soit dérivé de l'adresse de la couche de liaison du dispositif. Cela fait apparaître une variation de l'adresse de la couche de liaison dans l'adresse IPv6.

## Configuration des Adresses IPv6

Comme pour l'IPv4, l'adresse IPv6 peut être attribuée manuellement ou automatiquement. Cependant, IPv6 a deux types de configuration automatisée différents, DHCPv6 et *Stateless Address Autoconfiguration* (SLAAC).

La SLAAC est la plus simple des deux méthodes automatisées et elle est directement intégrée à la norme IPv6. Les messages utilisent le nouveau protocole *Neighbor Discovery Protocol* qui permet aux appareils de se retrouver et de demander des informations concernant un réseau. Ces informations sont envoyées par des routeurs et peuvent contenir des préfixes IPv6 que les appareils peuvent utiliser en les combinant avec un identificateur d'interface de leur choix, à condition que l'adresse résultante ne soit pas encore utilisée. Les appareils ne fournissent pas de retour d'information au routeur sur les adresses réelles qu'ils ont créées.

Le DHCPv6, d'autre part, est le DHCP mis à jour pour fonctionner avec les changements de l'IPv6. Il permet un contrôle plus fin des informations transmises aux clients, comme la possibilité de transmettre la même adresse au même client à chaque fois, et d'envoyer plus d'options au client que le SLAAC. Avec DHCPv6, les clients doivent obtenir le consentement explicite d'un serveur DHCP pour pouvoir utiliser une adresse.

La méthode pour attribuer manuellement une adresse IPv6 à une interface est la même qu'avec IPv4 :

```
$ sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev ens33
```

Pour vérifier que l'affectation a fonctionné, la même commande `ip addr show` est utilisée :

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
    inet6 2001:db8:0:abcd::7334/64 scope global
        valid_lft forever preferred_lft forever
```

Ici, nous voyons également l'adresse locale du lien `fe80::010c:29ff:fe33:3b25/64`.

Comme pour IPv4, la commande `ping` peut également être utilisée pour confirmer l'accessibilité des appareils via IPv6 :

```
$ ping 2001:db8:0:abcd::1
PING 2001:db8:0:abcd::1(2001:db8:0:abcd::1) 56 data bytes
64 bytes from 2001:db8:0:abcd::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=3 ttl=64 time=0.072 ms

--- 2001:db8:0:abcd::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 43ms
rtt min/avg/max/mdev = 0.030/0.047/0.072/0.018 ms
```

## NOTE

Sur certains systèmes Linux, le `ping` ne prend pas en charge IPv6. Ces systèmes fournissent à la place une commande `ping6` dédiée.

Pour vérifier à nouveau l'adresse du lien local, utilisez à nouveau le `ping`. Mais comme toutes les interfaces utilisent le préfixe `fe80::/64`, l'interface correcte doit être spécifiée avec l'adresse :

```
$ ping6 -c 1 fe80::010c:29ff:fe33:3b25%ens33
PING fe80::010c:29ff:fe33:3b25(fe80::010c:29ff:fe33:3b25) 56 data bytes
64 bytes from fe80::010c:29ff:fe33:3b25%ens33: icmp_seq=1 ttl=64 time=0.049 ms

--- fe80::010c:29ff:fe33:3b25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

## DNS

Les adresses IP sont difficiles à mémoriser et n'ont pas un facteur esthétique élevé si vous essayez de commercialiser un service ou un produit. C'est là que le *système des noms de domaine* entre en jeu. Dans sa forme la plus simple, le DNS est un annuaire téléphonique distribué qui associe des noms de domaine faciles à mémoriser, tels que `example.com`, à des adresses IP. Lorsqu'un utilisateur se rend sur un site web, par exemple, il saisit le nom d'hôte du DNS dans le cadre de l'URL. Le navigateur web envoie ensuite le nom DNS à l'un des résolveurs DNS configurés. Ce résolveur DNS trouvera à son tour l'adresse qui correspond au domaine. Le résolveur répond alors avec cette adresse et le navigateur web essaie d'atteindre le serveur web à cette adresse IP.

Les résolveurs que Linux utilise pour consulter les données DNS sont configurés dans le fichier de configuration `/etc/resolv.conf` :

```
$ cat /etc/resolv.conf
search lpi
nameserver 192.168.0.1
```

Lorsque le résolveur effectue une recherche de nom, il vérifie d'abord le fichier `/etc/hosts` pour voir s'il contient une adresse pour le nom demandé. Si c'est le cas, il renvoie cette adresse et ne contacte pas le DNS. Cela permet aux administrateurs de réseau de fournir une résolution de nom sans avoir à passer par l'effort de configuration d'un serveur DNS complet. Chaque ligne de ce fichier contient une adresse IP suivie d'un ou plusieurs noms :

```
127.0.0.1          localhost.localdomain localhost
::1               localhost.localdomain localhost
192.168.0.10      server
2001:db8:0:abcd::f server
```

Pour effectuer une recherche dans le DNS, utilisez la commande `host` :

```
$ host learning.lpi.org
learning.lpi.org has address 208.94.166.198
```

Des informations plus détaillées peuvent être obtenues en utilisant la commande `dig` :

```
$ dig learning.lpi.org
; <<>> DiG 9.14.3 <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 21525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2ac55879b1adef30a93013705d3306d2128571347df8eadf (bad)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.      550 IN  A      208.94.166.198

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sat Jul 20 14:20:21 EST 2019
;; MSG SIZE rcvd: 89
```

Ici, nous voyons également le nom des types d'enregistrements DNS, en l'occurrence A pour IPv4.

## Les Sockets

Un *socket* est un point d'extrémité de communication pour deux programmes qui se parlent. Si le socket est connecté sur un réseau, les programmes peuvent s'exécuter sur différents appareils, tels qu'un navigateur web fonctionnant sur l'ordinateur portable d'un utilisateur et un serveur web fonctionnant dans le centre de données d'une entreprise.

Il existe trois grands types de sockets :

### Sockets Unix

qui relie des processus s'exécutant sur le même appareil.

## Sockets UDP (User Datagram Protocol)

qui connectent des applications à l'aide d'un protocole rapide mais non résilient.

## Sockets TCP (Transmission Control Protocol)

qui sont plus fiables que les sockets UDP et qui, par exemple, confirment la réception des données.

Les sockets Unix ne peuvent connecter que des applications fonctionnant sur le même appareil. Les sockets TCP et UDP peuvent cependant se connecter sur un réseau. Le protocole TCP permet d'obtenir un flux de données qui arrive toujours dans l'ordre exact où il a été envoyé. UDP est plus *"tire et oublie"*; le paquet est envoyé mais sa livraison à l'autre extrémité n'est pas garantie. UDP n'a cependant pas la surcharge du TCP, ce qui le rend parfait pour les applications à faible latence telles que les jeux vidéo en ligne.

TCP et UDP utilisent tous deux des ports pour adresser plusieurs sockets sur la même adresse IP. Alors que le port source d'une connexion est généralement aléatoire, les ports cibles sont normalisés pour un service spécifique. HTTP est, par exemple, généralement hébergé sur le port 80, HTTPS est exécuté sur le port 443. SSH, un protocole permettant de se connecter en toute sécurité à un système Linux distant, écoute sur le port 22.

La commande `ss` permet à un administrateur d'enquêter sur toutes les sockets d'un ordinateur Linux. Elle affiche tout, de l'adresse source à l'adresse de destination en passant par le type. L'une de ses meilleures caractéristiques est l'utilisation de filtres permettant à un utilisateur de surveiller les sockets dans l'état de connexion qu'il souhaite. `ss` peut être lancée avec un ensemble d'options ainsi qu'une expression de filtre pour limiter les informations affichées.

Lorsqu'elle est exécutée sans aucune option, la commande affiche une liste de tous les sockets établis. L'utilisation de l'option `-p` inclut des informations sur le processus utilisant chaque socket. L'option `-s` affiche un résumé des sockets. Il y a beaucoup d'autres options disponibles pour cet outil, mais les dernières principales sont `-4` et `-6` pour réduire le protocole IP à IPv4 ou IPv6 respectivement, `-t` et `-u` permettent à l'administrateur de sélectionner les sockets TCP ou UDP et `-l` de n'afficher que les sockets qui écoutent les nouvelles connexions.

La commande suivante, par exemple, liste tous les sockets TCP actuellement utilisés :

```
$ ss -t
State      Recv-Q  Send-Q   Local Address:Port    Peer Address:Port
ESTAB      0        0       192.168.0.5:49412     192.168.0.1:https
ESTAB      0        0       192.168.0.5:37616     192.168.0.1:https
ESTAB      0        0       192.168.0.5:40114     192.168.0.1:https
ESTAB      0        0       192.168.0.5:54948     192.168.0.1:imap
```

...



## Exercices Guidés

1. Il est demandé à un ingénieur réseau d'attribuer deux adresses IP à l'interface `ens33` d'un hôte, une adresse IPv4 (`192.168.10.10/24`) et une adresse IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Quelles commandes doit-il entrer pour y parvenir ?

  

2. Quelles sont les adresses de la liste ci-dessous qui sont privées ?

192.168.10.1	
120.56.78.35	
172.16.57.47	
10.100.49.162	
200.120.42.6	

3. Quelle entrée ajouteriez-vous dans le fichier `hosts` pour attribuer `192.168.0.15` à `example.com` ?

4. Quel serait l'effet de la commande suivante ?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

## Exercices d'Exploration

1. Nommez le type d'enregistrement DNS utilisé pour servir les demandes suivantes :

- Données textuelles

- Recherche inversée d'adresses IP

- Un domaine qui n'a pas d'adresse propre et qui dépend d'un autre domaine pour cette information

- Serveur de courrier électronique

2. Linux a une fonction appelée *bridging*, que fait-elle et en quoi est-elle utile ?

3. Quelle option doit être fournie à la commande `ss` afin de visualiser tous les sockets UDP établies ?

4. Quelle commande affiche un résumé de tous les sockets fonctionnant sur un appareil Linux ?

5. La sortie suivante est générée par la commande de l'exercice précédent. Combien de sockets TCP et UDP sont actifs ?

```
Total: 978 (kernel 0)
TCP: 4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total      IP      IPv6
*      0          -          -
RAW    1          0          1
UDP    7          5          2
TCP    4          3          1
INET   12         8          4
FRAG   0          0          0
```

## Résumé

Ce sujet couvre la mise en réseau de votre ordinateur Linux. Nous avons d'abord découvert les différents niveaux de mise en réseau :

- La couche de liaison qui relie directement les appareils.
- La couche réseau qui assure le routage entre les réseaux et un espace d'adressage global.
- La couche application où les applications se connectent les unes aux autres.

Nous avons vu comment IPv4 et IPv6 sont utilisés pour adresser les ordinateurs individuels, et comment TCP et UDP énumèrent les sockets utilisés par les applications pour se connecter les unes aux autres. Nous avons également appris comment le DNS est utilisé pour résoudre les noms en adresses IP.

Commandes utilisées dans les exercices :

### **dig**

Interroge les informations du DNS et fournit des informations verbeuses sur les requêtes et les réponses du DNS.

### **host**

Interroge les informations du DNS et fournit un résultat condensé.

### **ip**

Configure la mise en réseau sous Linux, y compris les interfaces réseau, les adresses et le routage.

### **ping**

Teste la connectivité à un appareil distant.

### **ss**

Affiche les informations concernant les sockets.

## Réponses aux Exercices Guidés

1. Il est demandé à un ingénieur réseau d'attribuer deux adresses IP à l'interface `ens33` d'un hôte, une adresse IPv4 (`192.168.10.10/24`) et une adresse IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Quelles commandes doit-il entrer pour y parvenir ?

```
sudo ip addr add 192.168.10.10/24 dev ens33
sudo ip addr add 2001:0:0:abcd:0:8a2e:0370:7334/64 dev ens33
```

2. Quelles sont les adresses de la liste ci-dessous qui sont privées ?

192.168.10.1	X
120.56.78.35	
172.16.57.47	X
10.100.49.162	X
200.120.42.6	

3. Quelle entrée ajouteriez-vous dans le fichier `hosts` pour attribuer `192.168.0.15` à `example.com` ?

```
192.168.0.15 example.com
```

4. Quel serait l'effet de la commande suivante ?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Il ajouterait une route par défaut dans le tableau qui envoie tout le trafic IPv6 au routeur avec une adresse interne de `2001:db8:0:abcd::1`.

# Réponses aux Exercices d'Exploration

1. Nommez le type d'enregistrement DNS utilisé pour servir les demandes suivantes :

- Données textuelles

TXT

- Recherche inversée d'adresses IP

PTR

- Un domaine qui n'a pas d'adresse propre et qui dépend d'un autre domaine pour cette information

CNAME

- Serveur de courrier électronique

MX

2. Linux a une fonction appelée *bridging*, que fait-elle et en quoi est-elle utile ?

Un pont relie plusieurs interfaces de réseau. Toutes les interfaces connectées à un pont peuvent communiquer comme si elles étaient connectées au même réseau de couche de liaison : Tous les appareils utilisent des adresses IP provenant du même sous-réseau et n'ont pas besoin de routeur pour se connecter les uns aux autres.

3. Quelle option doit être fournie à la commande `ss` afin de visualiser tous les sockets UDP établies ?

L'option `-u` montre tous les sockets UDP établies.

4. Quelle commande affiche un résumé de tous les sockets fonctionnant sur un périphérique Linux ?

La commande `ss -s` affiche un résumé de tous les sockets

5. La sortie suivante est générée par la commande de l'exercice précédent. Combien de sockets TCP et UDP sont actifs ?

```
Total: 978 (kernel 0)
TCP:    4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0
```

	Transport	Total	IP	IPv6
*	0	-	-	
RAW	1	0	1	
UDP	7	5	2	
TCP	4	3	1	
INET		12	8	4
FRAG		0	0	0

11 sockets TCP et UDP sont actifs.



## **Sujet 5 : Sécurité et droits d'accès aux fichiers**



## 5.1 Sécurité élémentaire et identification des catégories d'utilisateurs

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 5.1

### Valeur

2

### Domaines de connaissance les plus importants

- Root et utilisateurs standards
- Utilisateurs système

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `/etc/passwd`, `/etc/shadow`, `/etc/group`
- `id`, `last`, `who`, `w`
- `sudo`, `su`





## 5.1 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	5 Sécurité et Permissions des Fichiers
<b>Objectif :</b>	5.1 Sécurité de Base et Identification des Types d'Utilisateurs
<b>Leçon :</b>	1 sur 1

### Introduction

Cette leçon se concentrera sur la terminologie de base des comptes, des contrôles d'accès et de la sécurité des systèmes Linux locaux, sur les outils de l'interface en ligne de commande (CLI) dans un système Linux pour les contrôles d'accès de sécurité de base et sur les fichiers de base pour prendre en charge les comptes d'utilisateurs et les groupes, y compris ceux utilisés pour l'escalade des privilèges élémentaires.

La sécurité de base des systèmes Linux est modelée sur les contrôles d'accès Unix qui, bien que vieux de près de cinquante ans, sont assez efficaces par rapport à certains systèmes d'exploitation grand public populaires de lignée beaucoup plus récente. Même certains autres systèmes d'exploitation populaires basés sur Unix ont tendance à “prendre des libertés” qui sont axées sur la “facilité d'accès”, alors que Linux ne le fait pas.

Les environnements et interfaces modernes de bureau Linux simplifient la création et la gestion des utilisateurs et automatisent souvent l'attribution des contrôles d'accès lorsqu'un utilisateur se connecte, par exemple, à l'écran, l'audio et les autres services ne requièrent pratiquement aucune intervention manuelle de l'administrateur système. Toutefois, il est important de comprendre les

concepts de base du système d'exploitation Linux sous-jacent.

## Les Comptes

La sécurité implique de nombreux concepts, l'un des plus courants étant le concept général de contrôle d'accès. Avant de pouvoir aborder les contrôles d'accès aux fichiers tels que la propriété et les permissions, il faut comprendre les concepts de base des *comptes* utilisateurs Linux, qui sont répartis en plusieurs types.

Chaque utilisateur d'un système Linux a un compte associé qui, outre les informations de connexion (comme le nom d'utilisateur et le mot de passe), définit également comment et où l'utilisateur peut interagir avec le système. Les privilèges et les contrôles d'accès définissent les "limites" à l'intérieur desquelles chaque utilisateur peut opérer.

## Les Identifiants (UIDs/GIDs)

Les *identificateurs d'utilisateur* et de *groupe* (UID/GID) sont les références de base, énumérées des comptes. Les premières versions étaient des nombres entiers limités à 16 bits (valeurs de 0 à 65535), mais les systèmes du 21<sup>e</sup> siècle prennent en charge les UID et GID de 64 bits. Les utilisateurs et les groupes sont énumérés indépendamment, de sorte que le même ID peut représenter à la fois un utilisateur et un groupe.

Chaque utilisateur dispose non seulement d'un UID, mais aussi d'un *GID primaire*. Le GID primaire d'un utilisateur peut être unique pour cet utilisateur seulement, et peut finir par ne pas être utilisé par d'autres utilisateurs. Toutefois, ce groupe peut également être un groupe partagé par de nombreux utilisateurs. En plus de ces groupes primaires, chaque utilisateur peut également être membre d'autres groupes.

Par défaut sur les systèmes Linux, chaque utilisateur est assigné à un groupe ayant le même nom que son nom d'utilisateur, et le même GID que son UID. Par exemple, créer un nouvel utilisateur nommé `newuser` et, par défaut, son groupe par défaut est également `newuser`.

## Le Compte du Super-utilisateur

Sous Linux, le compte du super-utilisateur est `root`, qui a toujours l'UID 0. Le super-utilisateur est parfois appelé *l'administrateur du système*, et a un accès et un contrôle illimités sur le système, y compris sur les autres utilisateurs.

Le groupe par défaut pour le super-utilisateur a le GID 0 et est également nommé `root`. Le répertoire personnel du super-utilisateur est un répertoire dédié de premier niveau, `/root`, accessible uniquement par l'utilisateur `root` lui-même.

## Comptes d'Utilisateur Standard

Tous les comptes autres que le compte `root` sont techniquement des comptes d'utilisateurs réguliers, mais sur un système Linux, le terme familier de *compte d'utilisateur* signifie souvent un compte d'utilisateur "régulier" (non privilégié). Ils ont généralement les propriétés suivantes, à quelques exceptions près :

- Les `UIDs` commencent à 1000 (4 chiffres), bien que certains anciens systèmes puissent commencer à 500.
- Un répertoire `home` défini, généralement un sous-répertoire de `/home`, en fonction de la configuration du site.
- Un `shell` de connexion défini. Sous Linux, le `shell` par défaut est généralement le *Bourne Again Shell* (`/bin/bash`), bien que d'autres `shells` puissent être disponibles.

Si un compte utilisateur ne dispose pas d'un `shell` valide dans ses attributs, l'utilisateur ne pourra pas ouvrir un `shell` interactif. Habituellement, `/sbin/nologin` est utilisé comme `shell` non valide. Cela peut être utile, si l'utilisateur n'est authentifié que pour des services autres que l'accès à la console ou au SSH, par exemple un accès FTP sécurisé (`sftp`) uniquement.

### NOTE

Pour éviter toute confusion, le terme *compte utilisateur* ne s'appliquera qu'aux comptes utilisateurs standards ou réguliers à l'avenir. Par exemple, le terme *compte système* sera utilisé pour expliquer un compte utilisateur Linux qui est du type compte utilisateur système.

## Comptes du Système

Les *comptes système* sont généralement pré-crés au moment de l'installation du système. Ils sont destinés aux utilitaires, programmes et services qui ne sont pas exécutés par le super-utilisateur. Dans un monde idéal, il s'agirait d'utilitaires du système d'exploitation.

Les comptes du système varient, mais leurs attributs comprennent :

- Les `UID` sont généralement inférieurs à 100 (2 chiffres) ou 500-1000 (3 chiffres).
- Soit *pas* répertoire personnel dédié, soit un répertoire qui n'est généralement pas sous `/home`.
- Pas de `shell` de connexion valide (généralement `/sbin/nologin`), à de rares exceptions près.

La plupart des comptes système sous Linux ne se connecteront jamais et n'ont pas besoin d'un `shell` défini dans leurs attributs. De nombreux processus détenus et exécutés par les comptes système sont intégrés dans leur propre environnement par la direction du système, et fonctionnent avec le compte système spécifié. Ces comptes ont généralement des privilèges limités

ou, le plus souvent, *aucun* privilège.

**NOTE**

Du point de vue du LPI Linux Essentials, les comptes système ont des UID <1000, avec des UID (et GID) à 2 ou 3 chiffres.

En général, les comptes du système *ne* doivent *pas* avoir de shell de connexion valide. Le contraire serait un problème de sécurité dans la plupart des cas.

## Comptes des Services

Les *comptes de service* sont généralement créés lorsque les services sont installés et configurés. Tout comme les comptes système, ils sont destinés aux installations, programmes et services qui ne s'exécuteront pas avec les privilèges du super-utilisateur.

Dans de nombreux documents, les comptes des systèmes et des services sont similaires et souvent interchangeables. Cela inclut l'emplacement des répertoires personnels qui se trouvent généralement en dehors de `/home`, s'ils sont définis (les comptes de service sont souvent plus susceptibles d'en avoir un, par rapport aux comptes système), et l'absence de shell de connexion valide. Bien qu'il n'y ait pas de définition stricte, la principale différence entre les comptes système et les comptes de service se résume à l'UID/GID.

### Compte du système

UID/GID <100 (2 chiffres) ou <500-1000 (3 chiffres)

### Compte de service

UID/GID >1000 (4+ chiffres), mais pas un compte utilisateur "standard" ou "régulier", un compte pour les services, avec un UID/GID >1000 (4+ chiffres)

Certaines distributions Linux ont encore des comptes de service réservés sous l'UID <100, et ceux-ci pourraient être considérés comme un compte système également, même s'ils ne sont pas créés lors de l'installation du système. Par exemple, sur les distributions Linux basées sur Fedora (y compris Red Hat), l'utilisateur du serveur web Apache a l'UID (et le GID) 48, ce qui est clairement un compte système, bien qu'il ait un répertoire personnel (généralement `/usr/share/httpd` ou `/var/www/html/`).

**NOTE**

Du point de vue de LPI Linux Essentials, les comptes système ont des UID <1000, et les comptes utilisateurs réguliers ont des UID > 1000. Comme les UID des comptes utilisateurs réguliers sont >1000, ces UID peuvent également inclure des comptes de services.

## Shells de Connexion et Répertoires Personnels

Certains comptes disposent d'un shell de connexion, tandis que d'autres n'en ont pas pour des raisons de sécurité car ils ne nécessitent pas d'accès interactif. Le shell de connexion par défaut sur la plupart des distributions Linux est le *Bourne Again Shell*, ou `bash`, mais d'autres shells peuvent être disponibles, comme le C Shell (`csh`), le Korn shell (`ksh`) ou le Z shell (`zsh`), pour n'en citer que quelques-uns.

Un utilisateur peut changer son shell de connexion en utilisant la commande `chsh`. Par défaut, la commande s'exécute en mode interactif, et affiche une invite demandant quel shell doit être utilisé. La réponse doit être le chemin complet vers le binaire de l'interpréteur de commandes, comme ci-dessous :

```
$ chsh
```

```
Changing the login shell for emma
Enter the new value, or press ENTER for the default
Login Shell [/bin/bash]: /usr/bin/zsh
```

Vous pouvez également exécuter la commande en mode non interactif, avec le paramètre `-s` suivi du chemin vers le binaire, comme ceci :

```
$ chsh -s /usr/bin/zsh
```

La plupart des comptes ont un répertoire personnel défini. Sous Linux, c'est généralement le seul endroit où ce compte utilisateur a un accès en écriture garanti, à quelques exceptions près (par exemple, les zones de système de fichiers temporaires). Toutefois, certains comptes sont délibérément configurés de manière à ne pas avoir d'accès en écriture, même à leur propre répertoire d'origine, pour des raisons de sécurité.

## Obtenir des Informations sur vos Utilisateurs

Lister les informations de base sur l'utilisateur est une pratique courante et quotidienne sur un système Linux. Dans certains cas, les utilisateurs devront changer d'utilisateur et augmenter leurs privilèges pour effectuer des tâches privilégiées.

Même les utilisateurs ont la possibilité de lister les attributs et d'y accéder à partir de la ligne de commande, en utilisant les commandes ci-dessous. Les informations de base dans un contexte limité ne constituent pas une opération privilégiée.

L'énumération des informations actuelles d'un utilisateur à la ligne de commande est aussi simple qu'une commande de deux lettres, `id`. La sortie variera en fonction de votre ID de connexion :

```
$ id
uid=1024(emma) gid=1024(emma) 1024(emma),20(games),groups=10240(netusers),20480(netadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Dans la liste précédente, l'utilisateur (emma) a des identifiants qui se répartissent comme suit :

- 1024 = ID utilisateur (UID), suivi du nom d'utilisateur (nom commun ou nom de connexion) entre parenthèses.
- 1024 = l'ID de *groupe primaire* (GID), suivi du nom de groupe (nom commun) entre parenthèses.
- Une liste de GIDs (noms de groupes) supplémentaires auxquels l'utilisateur appartient également.

La liste des derniers utilisateurs connectés au système est obtenue avec la commande `last` :

```
$ last
emma pts/3 ::1 Fri Jun 14 04:28 still logged in
reboot system boot 5.0.17-300.fc30. Fri Jun 14 04:03 still running
reboot system boot 5.0.17-300.fc30. Wed Jun 5 14:32 - 15:19 (00:46)
reboot system boot 5.0.17-300.fc30. Sat May 25 18:27 - 19:11 (00:43)
reboot system boot 5.0.16-100.fc28. Sat May 25 16:44 - 17:06 (00:21)
reboot system boot 5.0.9-100.fc28.x Sun May 12 14:32 - 14:46 (00:14)
root tty2 Fri May 10 21:55 - 21:55 (00:00)
...
```

Les informations énumérées dans les colonnes peuvent varier, mais certaines entrées notables de la liste précédente le sont :

- Un utilisateur (emma) s'est connecté via le réseau (pseudo TTY pts/3) et est toujours connecté.
- L'heure du démarrage en cours est indiquée, ainsi que le noyau. Dans l'exemple ci-dessus, environ 25 minutes avant que l'utilisateur ne se connecte.
- Le super-utilisateur (root) s'est connecté via une console virtuelle (TTY tty2), brièvement, à la mi-mai.

Une variante de la commande `last` est la commande `lastb`, qui liste toutes les dernières tentatives de connexion échouées.

Les commandes `who` et `w` ne listent que les connexions actives sur le système :

```
$ who
emma pts/3          2019-06-14 04:28 (::1)

$ w
 05:43:41 up  1:40,  1 user,  load average: 0.25, 0.53, 0.51
USER      TTY      LOGIN@  IDLE   JCPU   PCPU WHAT
emma pts/3    04:28   1:14m  0.04s  0.04s -bash
```

Les deux commandes listent certaines informations identiques. Par exemple, un utilisateur (`emma`) est connecté avec un pseudo appareil TTY (`pts/3`) et l'heure de connexion est `04:28`.

La commande `w` donne plus d'informations, dont les suivantes :

- L'heure actuelle et depuis combien de temps le système est démarré
- Combien d'utilisateurs sont connectés
- Les *moyennes de charge* des 1, 5 et 15 dernières minutes

Et les informations complémentaires pour chaque session d'utilisateur actif.

- Sélectionne, le temps total d'utilisation du CPU (`IDLE`, `JCPU` et `PCPU`)
- Le processus actuel (`-bash`). Le temps d'utilisation total de l'unité centrale de ce processus est le dernier élément (`PCPU`).

Les deux commandes disposent d'autres options permettant de lister diverses informations supplémentaires.

## Changement d'Utilisateur et Escalade des Privilèges

Dans un monde idéal, les utilisateurs n'auraient jamais besoin d'augmenter les privilèges pour accomplir leurs tâches. Le système fonctionnerait toujours "simplement" et tout serait configuré pour différents accès.

Heureusement pour nous, Linux fonctionne comme cela pour la plupart des utilisateurs qui ne sont pas administrateurs système, bien qu'ils suivent toujours le modèle de sécurité du *moindre privilège*.

Cependant, il existe des commandes qui permettent une escalade des privilèges en cas de besoin. Deux des plus importants sont `su` et `sudo`.

Sur la plupart des systèmes Linux actuels, la commande `su` n'est utilisée que pour obtenir les privilèges de root, qui est l'utilisateur par défaut si un nom d'utilisateur n'est pas spécifié après le nom de la commande. Bien qu'elle puisse être utilisée pour passer à un autre utilisateur, ce n'est pas une bonne pratique : les utilisateurs doivent se connecter depuis un autre système, via le réseau, ou une console ou un terminal physique sur le système.

```
emma ~$ su -
Password:
root ~#
```

Après avoir entré le mot de passe du super-utilisateur (root), l'utilisateur dispose d'un shell super-utilisateur (notez le `#` à la fin de l'invite de commande) et est, à toutes fins utiles, le super-utilisateur (root).

Le partage des mots de passe est une très mauvaise pratique de sécurité, il ne devrait donc y avoir que très peu, voire pas du tout, de besoin de la commande `su` dans un système Linux moderne.

Le symbole du dollar (\$) devrait terminer l'invite de la ligne de commande pour un shell utilisateur non privilégié, tandis que le symbole dièse (#) devrait terminer l'invite de la ligne de commande pour le shell du super-utilisateur (root). Il est fortement recommandé de *ne jamais* modifier le caractère final d'une invite par rapport à cette norme "universellement comprise", car cette nomenclature est utilisée dans les supports d'apprentissage, y compris ceux-ci.

Ne passez jamais au super-utilisateur (root) sans passer par le paramètre login shell (-). Sauf indication contraire explicite du système d'exploitation ou du fournisseur de logiciels lorsque `su` est requis, exécutez toujours `su -` avec des exceptions extrêmement limitées. Les environnements utilisateurs peuvent causer des changements de configuration et des problèmes indésirables lorsqu'ils sont utilisés en mode de privilèges complets en tant que super-utilisateur.

Quel est le principal problème lié à l'utilisation de `su` pour passer au super-utilisateur (root) ? Si la session d'un utilisateur normal a été compromise, le mot de passe du super-utilisateur (root) pourrait être saisi. C'est là qu'intervient le "Switch User Do" (ou "Superuser Do") :

```
$ cat /sys/devices/virtual/dmi/id/board_serial
cat: /sys/devices/virtual/dmi/id/board_serial: Permission denied

$ sudo cat /sys/devices/virtual/dmi/id/board_serial
[sudo] password for emma:
```



```
/6789ABC/
```

Dans la liste précédente, l'utilisateur tente de rechercher le numéro de série de sa carte système. Toutefois, la permission est refusée, car cette information est marquée comme étant privilégiée.

Cependant, en utilisant `sudo`, l'utilisateur entre son propre mot de passe pour s'authentifier. S'il a été autorisé dans la configuration de `sudoers` à exécuter cette commande avec privilège, avec les options autorisées, cela fonctionnera.

Par défaut, la première commande `sudo` autorisée authentifiera les commandes `sudo` suivantes pendant une période (très courte). Ceci est configurable par l'administrateur du système.

## Les Fichiers de Contrôle d'Accès

Presque tous les systèmes d'exploitation ont un ensemble d'endroits utilisés pour stocker les contrôles d'accès. Sous Linux, il s'agit généralement de fichiers texte situés dans le répertoire `/etc`, qui est l'endroit où les fichiers de configuration du système doivent être stockés. Par défaut, ce répertoire est lisible par tous les utilisateurs du système, mais il n'est accessible en écriture que par `root`.

Les principaux fichiers relatifs aux comptes utilisateurs, aux attributs et au contrôle d'accès sont les suivants

### `/etc/passwd`

Ce fichier contient des informations de base sur les utilisateurs du système, notamment l'UID et le GID, le répertoire personnel, le shell, etc. Malgré son nom, aucun mot de passe n'y est stocké.

### `/etc/group`

Ce fichier contient des informations de base sur tous les groupes d'utilisateurs du système, comme le nom du groupe, le GID et les membres.

### `/etc/shadow`

C'est là que les mots de passe des utilisateurs sont stockés. Ils sont hachés, pour des raisons de sécurité.

### `/etc/gshadow`

Ce fichier contient des informations plus détaillées sur les groupes, notamment un mot de passe haché qui permet aux utilisateurs de devenir temporairement membres du groupe, une

liste des utilisateurs qui peuvent devenir membres du groupe à un moment donné et une liste des administrateurs du groupe.

Ces fichiers ne sont pas conçus pour être édités directement et ne doivent jamais l'être. Cette leçon ne porte que sur les informations stockées dans ces fichiers, et non sur l'édition de ces fichiers.

Par défaut, chaque utilisateur peut entrer dans `/etc` et lire les fichiers `/etc/passwd` et `/etc/group`. Et aussi par défaut, aucun utilisateur, à l'exception de `root`, ne peut lire les fichiers `/etc/shadow` ou `/etc/gshadow`.

Il y a aussi des fichiers impliqués dans l'escalade des privilèges de base sur les systèmes Linux, comme les commandes `su` et `sudo`. Par défaut, ces fichiers ne sont accessibles qu'à l'utilisateur `root`.

### **`/etc/sudoers`**

Ce fichier contrôle qui peut utiliser la commande `sudo`, et comment.

### **`/etc/sudoers.d`**

Ce répertoire peut contenir des fichiers qui complètent les paramètres du fichier `sudoers`.

Du point de vue de l'examen LPI Linux Essentials, il suffit de connaître le chemin et le nom du fichier de configuration `sudo` par défaut, `/etc/sudoers`. Sa configuration dépasse le cadre de ces documents.

Même si `/etc/sudoers` est un fichier texte, il ne doit jamais être édité directement. Si des modifications de son contenu sont nécessaires, elles doivent être effectuées à l'aide de l'utilitaire `visudo`.

### **`/etc/passwd`**

Le fichier `/etc/passwd` est communément appelé "fichier de mots de passe". Chaque ligne contient plusieurs champs toujours délimités par deux points (:). Malgré son nom, le hachage de mot de passe à sens unique n'est pas actuellement stocké dans ce fichier.

La syntaxe typique pour une ligne de ce fichier est :

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
```

Où :

### USERNAME

Le nom d'utilisateur alias login (nom), comme `root`, `nobody`, `emma`.

### PASSWORD

Ancien emplacement du hachage du mot de passe. Presque toujours `x`, indiquant que le mot de passe est stocké dans le fichier `/etc/shadow`.

### UID

ID utilisateur (UID), comme `0`, `99`, `1024`.

### GID

ID de groupe par défaut (GID), comme `0`, `99`, `1024`.

### GECOS

Une liste CSV d'informations sur l'utilisateur comprenant le nom, la localisation, le numéro de téléphone. Par exemple : `Emma Smith,42 Douglas St,555.555.5555`

### HOMEDIR

Chemin d'accès au répertoire personnel de l'utilisateur, comme `/root`, `/home/emma`, etc.

### SHELL

Le shell par défaut pour cet utilisateur, comme `/bin/bash`, `/sbin/nologin`, `/bin/ksh`, etc.

Par exemple, la ligne suivante décrit l'utilisateur `emma` :

```
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

## Comprendre le Champ GECOS

Le champ GECOS contient trois (3) champs ou plus, délimités par une virgule (,), c'est-à-dire une liste de *Comma Separated Values* (CSV). Bien qu'il n'y ait pas de norme appliquée, les champs sont généralement dans l'ordre suivant :

```
NAME, LOCATION, CONTACT
```

Où :

## NAME

est le "Nom complet" de l'utilisateur, ou le "Nom du logiciel" dans le cas d'un compte de service.

## LOCATION

est généralement l'emplacement physique de l'utilisateur dans un bâtiment, le numéro de salle ou le service ou la personne de contact dans le cas d'un compte de service.

## CONTACT

contient des informations de contact telles que le numéro de téléphone du domicile ou du travail.

Les champs supplémentaires peuvent comprendre des informations de contact supplémentaires, telles qu'un numéro de domicile ou une adresse électronique. Pour modifier les informations du champ GECOS, utilisez la commande `chfn` et répondez aux questions, comme ci-dessous. Si aucun nom d'utilisateur n'est fourni après le nom de la commande, vous modifierez les informations concernant l'utilisateur courant :

```
$ chfn
```

```
Changing the user information for emma
Enter the new value, or press ENTER for the default
  Full Name: Emma Smith
  Room Number []: 42
  Work Phone []: 555.555.5555
  Home Phone []: 555.555.6666
```

## /etc/group

Le fichier `/etc/group` contient des champs toujours délimités par deux points (:), stockant des informations de base sur les groupes du système. Il est parfois appelé "fichier de groupe". La syntaxe de chaque ligne est :

```
NAME:PASSWORD:GID:MEMBERS
```

Où :

## NAME

est le nom du groupe, comme `root`, `users`, `emma`, etc.

## PASSWORD

l'ancien emplacement d'un hachage de mot de passe de groupe facultatif. Presque toujours `x`, indiquant que le mot de passe (s'il est défini) est stocké dans le fichier `/etc/gshadow`.

## GID

Group ID (GID), comme `0`, `99`, `1024`.

## MEMBERS

une liste de noms d'utilisateurs séparés par des virgules qui sont membres du groupe, comme `jsmith,emma`.

L'exemple ci-dessous montre une ligne contenant des informations sur le groupe `students` :

```
students:x:1023:jsmith,emma
```

L'utilisateur n'a pas besoin d'être listé dans le champ "membres" lorsque le groupe est le groupe principal d'un utilisateur. Si l'utilisateur est listé, alors c'est une information redondante, listé ou non cela n'a pas d'effet sur la fonctionnalité.

### NOTE

L'utilisation de mots de passe pour les groupes dépasse le cadre de cette section, mais si elle est définie, le hachage du mot de passe est stocké dans le fichier `/etc/gshadow`. Cela dépasse également le cadre de la présente section.

## /etc/shadow

Le tableau suivant énumère les attributs stockés dans le fichier `/etc/shadow`, communément appelé "fichier shadow". Le fichier contient des champs toujours délimités par deux points (:). Bien que le fichier comporte de nombreux champs, la plupart d'entre eux, à l'exception des deux premiers, dépassent le cadre de cette leçon.

La syntaxe de base pour une ligne de ce fichier est :

```
USERNAME:PASSWORD:LASTCHANGE:MINAGE:MAXAGE:WARN:INACTIVE:EXPDATE
```

Où :

## USERNAME

Le nom d'utilisateur (identique à `/etc/passwd`), comme `root`, `nobody`, `emma`.

**PASSWORD**

Un hachage à sens unique du mot de passe, y compris le *sel* précédent. Par exemple : `!!,$1$01234567$ABC..., $6$012345789ABCDEF$012....`

**LASTCHANGE**

Date du dernier changement de mot de passe en jours depuis l'“époque”, comme `17909`.

**MINAGE**

Âge minimum du mot de passe en jours.

**MAXAGE**

Âge maximum du mot de passe en jours.

**WARN**

Période d'avertissement avant l'expiration du mot de passe, en jours.

**INACTIVE**

Âge maximum du mot de passe après expiration, en jours.

**EXPDATE**

Date d'expiration du mot de passe, en jours depuis l'“époque”.

Dans l'exemple ci-dessous, vous pouvez voir un exemple d'entrée du fichier `/etc/shadow`. Notez que certaines valeurs, comme `INACTIVE` et `EXPDATE`, sont indéfinies.

```
emma:$6$nP532JDDogQYZF8I$bJFNh9eT1xpb9/n6pmj1Iwgu7hGjH/eytSdtbmvV0MlyTMFgBIXESFNUmTo9EGxxH1
OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

L'“époque” d'un système POSIX est minuit (0000), temps universel coordonné (UTC), le jeudi 1er janvier 1970. La plupart des dates et heures POSIX sont en secondes depuis l'“époque”, ou dans le cas du fichier `/etc/shadow`, en jours depuis l'“époque”.

**NOTE**

Le fichier `shadow` est conçu pour être uniquement lisible par le super-utilisateur et certains services d'authentification du système de base qui vérifient le hachage du mot de passe à sens unique lors de la connexion ou à un autre moment de l'authentification.

Bien qu'il existe différentes solutions d'authentification, la méthode élémentaire de stockage des mots de passe est la *fonction de hachage* à sens unique. Ainsi, le mot de passe n'est jamais stocké en clair sur un système, car la fonction de hachage n'est pas réversible. Vous pouvez transformer

un mot de passe en un hachage, mais (idéalement) il n'est pas possible de retransformer un hachage en un mot de passe.

Tout au plus, une méthode de force brute est nécessaire pour hacher toutes les combinaisons d'un mot de passe, jusqu'à ce que l'une d'entre elles corresponde. Pour éviter qu'un hachage de mot de passe ne soit craqué sur un système, les systèmes Linux utilisent un "sel" aléatoire sur chaque hachage de mot de passe pour un utilisateur. Ainsi, le hachage du mot de passe d'un utilisateur sur un système Linux ne sera généralement pas le même que sur un autre système Linux, même si le mot de passe est identique.

Dans le fichier `/etc/shadow`, le mot de passe peut prendre plusieurs formes. Ces formes sont généralement les suivantes :

**!!**

Cela signifie un compte "désactivé" (aucune authentification possible), sans hachage de mot de passe.

**!\$1\$01234567\$ABC...**

Un compte "désactivé" (en raison du point d'exclamation initial), avec une fonction de hachage préalable, un sel de hachage et une chaîne de hachage stockés.

**\$1\$0123456789ABC\$012...**

Un compte "activé", avec une fonction de hachage, un sel de hachage et une chaîne de hachage stockée.

La fonction de hachage, le sel de hachage et la chaîne de hachage sont précédés et délimités par un symbole de dollar (\$). La longueur du sel de hachage doit être comprise entre huit et seize (8-16) caractères. Voici des exemples des trois les plus courants :

**\$1\$01234567\$ABC...**

Une fonction de hachage du MD5 (1), avec un exemple de longueur de hachage de huit.

**\$5\$01234567ABCD\$012...**

Une fonction de hachage de SHA256 (5), avec un exemple de longueur de hachage de douze.

**\$6\$01234567ABCD\$012...**

Une fonction de hachage de SHA512 (6), avec un exemple de longueur de hachage de douze.

#### NOTE

La fonction de hachage du MD5 est considérée comme peu sûre sur le plan cryptographique avec le niveau actuel (années 2010 et suivantes) de performances des ASIC et même des SIMD informatiques générales. Par exemple, les normes

féderales américaines de traitement de l'information (FIPS) ne permettent pas l'utilisation du MD5 pour des fonctions cryptographiques, seulement pour des aspects très limités de la validation, mais pas pour l'intégrité des signatures numériques ou à des fins similaires.

Du point de vue des objectifs et de l'examen des LPI Linux Essentials, il suffit de comprendre que le hachage du mot de passe d'un utilisateur local n'est stocké que dans le fichier `/etc/shadow` que seuls les services d'authentification peuvent lire, ou que le super utilisateur peut modifier via d'autres commandes.



## Exercices Guidés

1. Considérez la sortie suivante de la commande `id` :

```
$ id emma
uid=1000(emma) gid=1000(emma)
groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
```

Dans quels fichiers sont stockés les attributs suivants ?

UID et GID	
Groupes	

- En outre, dans quel fichier le mot de passe de l'utilisateur est-il stocké ?

2. Parmi les types de cryptographie suivants, lequel est utilisé par défaut pour stocker les mots de passe localement sur un système Linux ?

- Asymétrique
- Hachage à sens unique
- Symétrique
- ROT13

3. Si un compte a un User ID (UID) inférieur à 1000, de quel type de compte s'agit-il ?

4. Comment obtenir une liste des connexions actives dans votre système, ainsi qu'un décompte de celles-ci ?

5. En utilisant la commande `grep`, nous avons obtenu le résultat ci-dessous avec des informations sur l'utilisateur `emma`.

```
$ grep emma /etc/passwd
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
```

Remplissez les blancs du tableau avec les informations appropriées en utilisant la sortie de la commande précédente.

Nom d'utilisateur	
Mot de passe	
UID	
GID primaire	
GECOS	
Répertoire personnel	
Shell	

## Exercices d'Exploration

1. Comparez les résultats de `last` à `w` et `who`. Quels sont les détails qui manquent à chacune des commandes comparées les unes aux autres ?

2. Essayez les commandes `who` et `w -his`.

- Quelles informations ont été supprimées de la sortie de la commande `w` avec les options “no header” (`-h`) et “short” (`-s`) ?

3. Quelles informations ont été ajoutées dans la sortie de la commande `w` avec l'option “ip address” (`-i`) ?

4. Quel est le fichier qui stocke le hachage du mot de passe à sens unique d'un compte utilisateur ?

5. Quel fichier contient la liste des groupes dont un compte utilisateur est membre ? Quelle logique pourrait être utilisée pour dresser la liste des groupes dont un compte utilisateur est membre ?

6. Un ou plusieurs (1+) des fichiers suivants ne sont pas lisibles par les utilisateurs réguliers, non privilégiés, par défaut. Lesquels ?

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`

7. Comment changer le shell de connexion de l'utilisateur actuel pour le Korn Shell (`/usr/bin/ksh`) en mode non-interactif ?

8. Pourquoi le répertoire d'origine de l'utilisateur `root` n'est-il pas placé dans le répertoire `/home` ?



## Résumé

Dans cette leçon, nous avons découvert les bases de données des utilisateurs et des groupes de Linux. Nous avons appris les propriétés les plus importantes des utilisateurs et des groupes, y compris leurs noms et leurs identifiants numériques. Nous avons également étudié le fonctionnement du hachage des mots de passe sous Linux et la manière dont les utilisateurs sont affectés à des groupes.

Toutes ces informations sont stockées dans les quatre fichiers suivants, qui fournissent les contrôles d'accès de sécurité locaux les plus élémentaires sur un système Linux :

### **/etc/passwd**

Tous les attributs POSIX du compte utilisateur local du système, autres que le hachage du mot de passe, lisibles par tous.

### **/etc/group**

Tous les attributs POSIX des comptes de groupe système-local, lisibles par tous.

### **/etc/shadow**

Tous les hashes de mots de passe des utilisateurs locaux du système (et les informations d'expiration), illisibles par n'importe qui (seulement certains processus).

### **/etc/sudoers**

Toutes les informations relatives à l'escalade des privilèges au niveau du système local/autorisable par la commande `sudo`.

Les commandes suivantes ont été abordées dans cette leçon :

### **id**

Liste les identifiants réels (ou effectifs) des utilisateurs et des groupes.

### **last**

Liste des utilisateurs qui se sont connectés en dernier.

### **who**

Liste des utilisateurs qui sont actuellement connectés.

### **w**

Semblable à `who` mais avec un contexte supplémentaire.

**su**

Change d'utilisateur avec un shell de connexion, ou exécute des commandes comme cet utilisateur, en lui transmettant son mot de passe.

**sudo**

Switch User (ou Superuser) Do : si l'utilisateur actuel y a droit, il doit entrer son propre mot de passe (si nécessaire) afin de soulever le privilège.

**chsh**

Change le shell d'un utilisateur.

**chfn**

Modifie les informations de l'utilisateur sur le champ GECOS.

# Réponses aux Exercices Guidés

1. Considérez la sortie suivante de la commande `id` :

```
$ id emma
uid=1000(emma) gid=1000(emma)
groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
```

Dans quels fichiers sont stockés les attributs suivants ?

UID et GID	/etc/passwd
Groupes	/etc/group

- En outre, dans quel fichier le mot de passe de l'utilisateur est-il stocké ?

Le mot de passe utilisateur haché est stocké dans `/etc/shadow`.

2. Parmi les types de cryptographie suivants, lequel est utilisé par défaut pour stocker les mots de passe localement sur un système Linux ?

Par défaut, un hachage à sens unique est utilisé pour stocker les mots de passe.

3. Si un compte a un User ID (UID) inférieur à 1000, de quel type de compte s'agit-il ?

Les comptes dont l'UID est inférieur à 1000 sont généralement des comptes système.

4. Comment obtenir une liste des connexions actives dans votre système, ainsi qu'un décompte de celles-ci ?

Utilisez la commande `w`. Outre une liste de toutes les connexions actives, elle affichera également des informations telles que le nombre d'utilisateurs connectés, la charge du système et le temps de fonctionnement.

5. En utilisant la commande `grep`, nous avons obtenu le résultat ci-dessous avec des informations sur l'utilisateur `emma`.

```
$ grep emma /etc/passwd
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
```

Remplissez les blancs du tableau avec les informations appropriées en utilisant la sortie de la commande précédente.

Nom d'utilisateur	emma
Mot de passe	x doit toujours être x pour une connexion d'utilisateur valide et active
UID	1000
GID primaire	1000
GECOS	Emma Smith,42 Douglas St,555.555.5555
Répertoire personnel	/home/emma
Shell	/bin/ksh



## Réponses aux Exercices d'Exploration

1. Comparez les résultats de `last` à `w` et `who`. Quels sont les détails qui manquent à chacune des commandes comparées les unes aux autres ?

Les outils `w` et `who` n'affichent que les utilisateurs actuels connectés au système, tandis que `last` affiche également les utilisateurs qui se sont déconnectés. La commande `w` liste l'utilisation du système, tandis que `who` ne le fait pas.

2. Essayez les commandes `who` et `w -his`.
3. Quelles informations ont été supprimées de la sortie de la commande `w` avec les options "no header" (`-h`) et "short" (`-s`) ?

L'en-tête n'est pas affiché, ce qui est utile pour l'analyse, l'heure de connexion et les informations sélectionnées relatives au CPU ne sont pas indiquées, respectivement.

- Quelles informations ont été ajoutées dans la sortie de la commande `w` avec l'option "ip address" (`-i`) ?

Cela permet d'afficher l'adresse IP, au lieu de tenter une résolution DNS, en affichant le nom d'hôte. Cette option permet de mieux correspondre à la sortie par défaut de la commande `last`.

4. Quel est le fichier qui stocke le hachage du mot de passe à sens unique d'un compte utilisateur ?

Le fichier `/etc/shadow` stocke le hachage du mot de passe à sens unique d'un compte utilisateur, puisqu'il n'est pas lisible par un compte utilisateur normal, non privilégié, contrairement au fichier `/etc/passwd`.

5. Quel fichier contient la liste des groupes dont un compte utilisateur est membre ? Quelle logique pourrait être utilisée pour dresser la liste des groupes dont un compte utilisateur est membre ?

Le fichier `/etc/group` contient une liste CSV des noms d'utilisateurs dans le dernier champ, "members", de n'importe quelle ligne d'un groupe.

Toute ligne du fichier `/etc/group` où l'utilisateur est listé dans le dernier champ, "members", signifierait que l'utilisateur est membre de ce groupe en supposant qu'il est correctement formaté (délimité par CSV). En outre, l'appartenance principale de l'utilisateur à un groupe dans le fichier `/etc/passwd` aura également une entrée correspondante dans le fichier `/etc/group` pour le nom du groupe et le GID.

6. Un ou plusieurs (1+) des fichiers suivants ne sont pas lisibles par les utilisateurs réguliers, non privilégiés, par défaut. Lesquels ?

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`

Les fichiers `/etc/shadow` et `/etc/sudoers` ne sont pas lisibles par défaut, sauf par certains services ou le super-utilisateur. Ils seront personnalisés, en fonction des systèmes et des noms d'utilisateur utilisés dans le laboratoire.

7. Comment changer le shell de connexion de l'utilisateur actuel pour le Korn Shell (`/usr/bin/ksh`) en mode non-interactif ?

```
$ chsh -s /usr/bin/ksh
```

8. Pourquoi le répertoire personnel de l'utilisateur `root` n'est-il pas placé dans le répertoire `/home` ?

Parce que le compte `root` est nécessaire pour dépanner et corriger les erreurs, qui peuvent inclure des problèmes de système de fichiers liés au répertoire `/home`. Dans ce cas, le compte `root` doit être pleinement fonctionnel même si le système de fichiers `/home` n'est pas encore disponible.



## 5.2 Création des utilisateurs et des groupes

### Référence aux objectifs de LPI

Linux Essentials version 1.6, Exam 010, Objective 5.2

### Valeur

2

### Domaines de connaissance les plus importants

- Commandes de création d'utilisateurs et de groupes
- Identification des utilisateurs

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/skel/`
- `useradd`, `groupadd`
- `passwd`



## 5.2 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	5 Sécurité et Permissions des Fichiers
<b>Objectif :</b>	5.2 Création d'Utilisateurs et de Groupes
<b>Leçon:</b>	1 sur 1

### Introduction

La gestion des utilisateurs et des groupes sur une machine Linux est l'un des aspects clés de l'administration système. En fait, Linux est un système d'exploitation multi-utilisateurs dans lequel plusieurs utilisateurs peuvent utiliser la même machine en même temps.

Les informations sur les utilisateurs et les groupes sont stockées dans quatre fichiers dans l'arborescence du répertoire `/etc/` :

#### **`/etc/passwd`**

un fichier de sept champs délimités par des deux points, contenant des informations de base sur les utilisateurs

#### **`/etc/group`**

un fichier de quatre champs délimités par des deux points, contenant des informations de base sur les groupes

#### **`/etc/shadow`**

un fichier de neuf champs délimités par des deux points, contenant les mots de passe hachés

des utilisateurs

### **/etc/gshadow**

un fichier de quatre champs délimités par des deux points, contenant des mots de passe de groupe hachés

Tous ces fichiers sont mis à jour par une suite d'outils en ligne de commande pour la gestion des utilisateurs et des groupes, dont nous parlerons plus loin dans cette leçon. Ils peuvent également être gérés par des applications graphiques, spécifiques à chaque distribution Linux, qui fournissent des interfaces de gestion plus simples et plus immédiates.

Même si les fichiers sont en texte clair, ne les modifiez pas directement. Utilisez toujours les outils fournis avec votre distribution à cette fin.

## **Le Fichier /etc/passwd**

/etc/passwd est un fichier lisible par tout le monde qui contient une liste d'utilisateurs, chacun sur une ligne séparée :

```
frank:x:1001:1001:./home/frank:/bin/bash
```

Chaque ligne se compose de sept champs délimités par des deux points :

### **Nom d'utilisateur**

Le nom utilisé lorsque l'utilisateur se connecte au système.

### **Mot de passe**

Le mot de passe haché (ou un `x` si des mots de passe *shadow* sont utilisés).

### **User ID (UID)**

Le numéro d'identification attribué à l'utilisateur dans le système.

### **Group ID (GID)**

Le numéro du groupe principal de l'utilisateur dans le système.

### **GECOS**

Un champ de commentaire optionnel, qui est utilisé pour ajouter des informations supplémentaires sur l'utilisateur (comme le nom complet). Le champ peut contenir plusieurs entrées séparées par des virgules.

## Répertoire personnel

Le chemin absolu du répertoire personnel de l'utilisateur.

## Shell

Le chemin absolu du programme qui est automatiquement lancé lorsque l'utilisateur se connecte au système (généralement un shell interactif tel que `/bin/bash`).

## Le Fichier `/etc/group`

`/etc/group` est un fichier lisible par tout le monde qui contient une liste de groupes, chacun sur une ligne séparée :

```
developer:x:1002:
```

Chaque ligne est constituée de quatre champs délimités par des deux points :

### Nom du groupe

Le nom du groupe.

### Mot de passe du groupe

Le mot de passe haché du groupe (ou un `x` si des mots de passe *shadow* sont utilisés).

### Group ID (GID)

Le numéro d'identification attribué au groupe dans le système.

### Liste des membres

Une liste, délimitée par des virgules, des utilisateurs appartenant au groupe, à l'exception de ceux pour lesquels il s'agit du groupe principal.

## Le Fichier `/etc/shadow`

`/etc/shadow` est un fichier lisible uniquement par root et les utilisateurs ayant des privilèges de root, il contient les mots de passe hachés des utilisateurs, chacun sur une ligne séparée :

```
frank:$6$i9gjM4Md4Mue1ZCd$7jJa8Cd2bbADFH4dwtfvTvJLOYCCCBf/.jYbK1IMYx7Wh4fErXcc2xQVU2N1gb97yI  
YaiqH.jjJammzof2Jfr/:18029:0:99999:7:::
```

Chaque ligne est constituée de neuf champs délimités par des deux points :

## Nom d'utilisateur

Le nom utilisé lorsque l'utilisateur se connecte au système.

## Mot de passe haché

Le mot de passe haché de l'utilisateur (si la valeur est `!`, le compte est verrouillé).

## Date du dernier changement de mot de passe

La date du dernier changement de mot de passe, en nombre de jours depuis le 01/01/1970. Une valeur de `0` signifie que l'utilisateur doit changer le mot de passe lors du prochain accès.

## Âge minimum du mot de passe

Le nombre minimum de jours, après un changement de mot de passe, qui doit s'écouler avant que l'utilisateur soit autorisé à changer à nouveau le mot de passe.

## Âge maximum du mot de passe

Le nombre maximum de jours qui doivent s'écouler avant qu'un changement de mot de passe soit nécessaire.

## Période d'avertissement du mot de passe

Le nombre de jours, avant l'expiration du mot de passe, pendant lesquels l'utilisateur est averti que le mot de passe doit être modifié.

## Période d'inactivité du mot de passe

Le nombre de jours après l'expiration d'un mot de passe pendant lesquels l'utilisateur doit le mettre à jour. Après cette période, si l'utilisateur ne modifie pas le mot de passe, le compte sera désactivé.

## Date d'expiration du compte

La date, en nombre de jours depuis le 01/01/1970, à laquelle le compte utilisateur sera désactivé. Un champ vide signifie que le compte utilisateur n'expirera jamais.

## Un champ réservé

Un champ qui est réservé pour un usage futur.

## Le Fichier `/etc/gshadow`

`/etc/gshadow` est un fichier lisible uniquement par root et par les utilisateurs disposant de privilèges root, qui contient des mots de passe hachés pour les groupes, chacun sur une ligne séparée :

```
developer:$6$7QUIhUX1Wd06$H7k0YgsboLkDseFHpk04lwAtweSUQHipoXIgo83QNDxYtYwgmZTCU0qSCuCkErmyR2
63rvHiLctZVDR7Ya9Ai1::
```

Chaque ligne est constituée de quatre champs délimités par des deux points :

### Nom du groupe

Le nom du groupe.

### Mot de passe haché

Le mot de passe haché du groupe (il est utilisé lorsqu'un utilisateur, qui n'est pas membre du groupe, veut rejoindre le groupe en utilisant la commande `newgrp` si le mot de passe commence par `!` personne n'est autorisé à accéder au groupe avec `newgrp`).

### Administrateurs du groupe

Une liste, délimitée par des virgules, des administrateurs du groupe (ils peuvent changer le mot de passe du groupe et peuvent ajouter ou supprimer des membres du groupe avec la commande `gpasswd`).

### Membres du groupe

Une liste des membres du groupe, délimitée par des virgules.

Maintenant que nous avons vu où sont stockées les informations sur les utilisateurs et les groupes, parlons des principaux outils en ligne de commande pour mettre à jour ces fichiers.

## Ajout et Suppression de Comptes Utilisateurs

Sous Linux, vous ajoutez un nouveau compte utilisateur avec la commande `useradd`, et vous supprimez un compte utilisateur avec la commande `userdel`.

Si vous souhaitez créer un nouveau compte utilisateur nommé `frank` avec un paramètre par défaut, vous pouvez exécuter ce qui suit :

```
# useradd frank
```

Après avoir créé le nouvel utilisateur, vous pouvez définir un mot de passe à l'aide de `passwd` :

```
# passwd frank
Changing password for user frank.
New UNIX password:
```



```
Retype new UNIX password:  
passwd: all authentication tokens updated successfully.
```

Ces deux commandes requièrent les autorisations de root. Lorsque vous exécutez la commande `useradd`, les informations sur l'utilisateur et le groupe stockées dans les bases de données de mots de passe et de groupes sont mises à jour pour le compte utilisateur nouvellement créé et, si cela est spécifié, le répertoire personnel du nouvel utilisateur est créé ainsi qu'un groupe portant le même nom que le compte utilisateur.

N'oubliez pas que vous pouvez toujours utiliser l'utilitaire `grep` pour filtrer les bases de données de mots de passe et de groupes, en n'affichant que l'entrée qui se réfère à un utilisateur ou un groupe spécifique. Pour l'exemple ci-dessus, vous pouvez utiliser

```
cat /etc/passwd | grep frank
```

ou

```
grep frank /etc/passwd
```

pour voir les informations de base sur le compte `frank` nouvellement créé.

Les options les plus importantes qui s'appliquent à la commande `useradd` sont les suivantes :

**-c**

Crée un nouveau compte utilisateur avec des commentaires personnalisés (par exemple le nom complet).

**-d**

Crée un nouveau compte utilisateur avec un répertoire personnel défini.

**-e**

Crée un nouveau compte utilisateur en fixant une date précise à laquelle il sera désactivé.

**-f**

Crée un nouveau compte utilisateur en fixant le nombre de jours après l'expiration du mot de passe pendant lesquels l'utilisateur doit mettre à jour son mot de passe.

**-g**

Crée un nouveau compte utilisateur avec un GID spécifique

**-G**

Crée un nouveau compte utilisateur en l'ajoutant à plusieurs groupes secondaires.

**-m**

Crée un nouveau compte utilisateur avec son répertoire personnel.

**-M**

Crée un nouveau compte utilisateur sans son répertoire personnel.

**-s**

Crée un nouveau compte utilisateur avec un shell de connexion spécifique.

**-u**

Créer un nouveau compte utilisateur avec un UID spécifique.

Une fois le nouveau compte utilisateur créé, vous pouvez utiliser les commandes `id` et `groups` pour connaître son UID, son GID et les groupes auxquels il appartient.

```
# id frank
uid=1000(frank) gid=1000(frank) groups=1000(frank)
# groups frank
frank : frank
```

N'oubliez pas de vérifier et éventuellement de modifier le fichier `/etc/login.defs`, qui définit les paramètres de configuration qui contrôlent la création des utilisateurs et des groupes. Par exemple, vous pouvez définir la plage d'UID et de GID qui peuvent être attribués aux nouveaux comptes utilisateurs et de groupes, préciser qu'il n'est pas nécessaire d'utiliser l'option `-m` pour créer le répertoire personnel du nouvel utilisateur et si le système doit automatiquement créer un nouveau groupe pour chaque nouvel utilisateur.

Si vous souhaitez supprimer un compte utilisateur, vous pouvez utiliser la commande `userdel`. Cette commande permet notamment de mettre à jour les informations stockées dans les bases de données des comptes, en supprimant toutes les entrées se rapportant à l'utilisateur spécifié. L'option `-r` supprime également le répertoire personnel de l'utilisateur et tout son contenu, ainsi que l'ensemble des courriers électroniques de l'utilisateur. Les autres fichiers, situés ailleurs, doivent être recherchés et supprimés manuellement.

```
# userdel -r frank
```

Comme auparavant, vous avez besoin de la permission de root pour supprimer des comptes utilisateurs.

## Le Répertoire Squelette

Lorsque vous ajoutez un nouveau compte utilisateur, même en créant son répertoire personnel, le répertoire personnel nouvellement créé est rempli de fichiers et de dossiers qui sont copiés à partir du répertoire squelette (par défaut `/etc/skel`). L'idée est simple : un administrateur système veut ajouter de nouveaux utilisateurs ayant les mêmes fichiers et répertoires dans leur répertoire personnel. Par conséquent, si vous souhaitez personnaliser les fichiers et les dossiers qui sont créés automatiquement dans le répertoire personnel des nouveaux comptes utilisateurs, vous devez ajouter ces nouveaux fichiers et dossiers au répertoire squelette.

Notez que les fichiers de profil qui se trouvent généralement dans le répertoire du squelette sont des fichiers cachés. Par conséquent, si vous voulez lister tous les fichiers et dossiers du répertoire squelette, qui seront copiés dans le répertoire personnel des utilisateurs nouvellement créés, vous devez utiliser la commande `ls -Al`.

## Ajout et Suppression de Groupes

Quant à la gestion des groupes, vous pouvez ajouter ou supprimer des groupes à l'aide des commandes `groupadd` et `groupdel`.

Si vous souhaitez créer un nouveau groupe nommé `developer`, vous pouvez exécuter la commande suivante en tant que root :

```
# groupadd -g 1090 developer
```

L'option `-g` de cette commande crée un groupe avec un GID spécifique.

Si vous souhaitez supprimer le groupe `developer`, vous pouvez lancer la commande suivante :

```
# groupdel developer
```

N'oubliez pas que lorsque vous ajoutez un nouveau compte utilisateur, le groupe primaire et les groupes secondaires auxquels il appartient doivent exister avant de lancer la commande `useradd`. De même, vous ne pouvez pas supprimer un groupe s'il s'agit du groupe primaire

d'un compte utilisateur.

## La Commande `passwd`

Cette commande est principalement utilisée pour changer le mot de passe d'un utilisateur. Tout utilisateur peut changer son mot de passe, mais seul root peut changer le mot de passe de n'importe quel utilisateur.

Selon l'option de `passwd` utilisée, vous pouvez contrôler des aspects spécifiques du vieillissement des mots de passe :

### **-d**

Supprime le mot de passe d'un compte utilisateur (ce qui désactive l'utilisateur).

### **-e**

Force le compte utilisateur à changer le mot de passe.

### **-l**

Verrouille le compte de l'utilisateur (le mot de passe haché est précédé d'un point d'exclamation).

### **-u**

Déverrouille le compte utilisateur (il supprime le point d'exclamation).

### **-S**

Produire des informations sur le statut du mot de passe pour un compte spécifique.

Ces options ne sont disponibles que pour root. Pour voir la liste complète des options, reportez-vous aux pages de manuel.

## Exercices Guidés

1. Pour chacune des entrées suivantes, indiquez le fichier auquel elle se rapporte :

- `developer:x:1010:frank,grace,dave`

- `root:x:0:0:root:/root:/bin/bash`

- `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

- `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

- `staff:!:dave:carol,emma`

2. Observez les résultats suivants pour répondre aux sept questions suivantes :

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbCdEfGhIjKlMnOpQrStu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

- Quels sont l'ID utilisateur (UID) et l'ID groupe (GID) de `carol` ?

- Quels sont les shells prévus pour `dave` et `henry` ?

- 
- Quel est le nom du groupe primaire de `henry` ?

- 
- Quels sont les membres du groupe `web_developer` ? Quels sont les administrateurs de ce groupe ?

- 
- Quel utilisateur ne peut pas se connecter au système ?

- 
- Quel utilisateur doit changer de mot de passe la prochaine fois qu'il se connectera au système ?

- 
- Combien de jours doivent s'écouler avant qu'un changement de mot de passe ne soit nécessaire pour `carol` ?
-

## Exercices d'Exploration

1. En tant que root, exécutez la commande `useradd -m dave` pour ajouter un nouveau compte utilisateur. Quelles sont les opérations effectuées par cette commande ? Supposons que `CREATE_HOME` et `USERGROUPS_ENAB` dans `/etc/login.defs` soient définis sur `yes`.

2. Maintenant que vous avez créé le compte `dave`, cet utilisateur peut-il se connecter au système ?

3. Identifier l'ID utilisateur (UID) et l'ID groupe (GID) de `dave` et de tous les membres du groupe `dave`.

4. Créez les groupes `sys_admin`, `web_admin` et `db_admin` et identifiez leurs ID de groupe (GID).

5. Ajoutez un nouveau compte utilisateur nommé `carol` avec l'UID 1035 et définir `sys_admin` comme son groupe primaire et `web_admin` et `db_admin` comme ses groupes secondaires.

6. Supprimez les comptes utilisateurs `dave` et `carol` et les groupes `sys_admin`, `web_admin` et `db_admin` que vous avez créés précédemment.

7. Exécutez la commande `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` et décrivez la sortie qu'elle vous donne en termes de permissions de fichiers. Lesquels de ces quatre fichiers sont ombragés (*shadowed*) pour des raisons de sécurité ? Supposons que votre système utilise des mots de passe "shadow".

8. Exécutez la commande `ls -l /usr/bin/passwd`. Quel bit spécial est défini et quelle est sa signification ?

# Résumé

Dans cette leçon, vous avez appris :

- Les bases de la gestion des utilisateurs et des groupes sous Linux
- À gérer les informations sur les utilisateurs et les groupes stockées dans les bases de données de mots de passe et de groupes
- À maintenir le répertoire squelette
- À ajouter et supprimer des comptes utilisateurs
- À ajouter et supprimer des comptes de groupes
- À changer le mot de passe des comptes utilisateurs

Les commandes suivantes ont été abordées dans cette leçon :

## **useradd**

Crée un nouveau compte utilisateur.

## **groupadd**

Crée un nouveau compte de groupe.

## **userdel**

Supprime un compte utilisateur.

## **groupdel**

Supprime un compte de groupe.

## **passwd**

Modifie le mot de passe des comptes utilisateurs et contrôler tous les aspects du vieillissement des mots de passe.



## Réponses aux Exercices Guidés

1. Pour chacune des entrées suivantes, indiquez le fichier auquel elle se rapporte :

- `developer:x:1010:frank,grace,dave`

`/etc/group`

- `root:x:0:0:root:/root:/bin/bash`

`/etc/passwd`

- `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

`/etc/shadow`

- `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

`/etc/passwd`

- `staff:!:dave:carol,emma`

`/etc/gshadow`

2. Observez les résultats suivants pour répondre aux sept questions suivantes :

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbCdEfGhIjKlMnOpQrStu:18015:0:60:7:::
henry:!!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!!:
```

- Quels sont l'ID utilisateur (UID) et l'ID groupe (GID) de `carol` ?

L'UID est 1051 et le GID est 1015 (les troisième et quatrième champs de `/etc/passwd`).

- Quels shells sont prévus pour `dave` et `henry` ?

`dave` utilise `/bin/bash` et `henry` utilise `/bin/tcsh` (le septième champ dans `/etc/passwd`).

- Quel est le nom du groupe primaire de `henry` ?

Le nom du groupe est `web_admin` (le premier champ dans `/etc/group`).

- Quels sont les membres du groupe des `web_developer` ? Quels sont les administrateurs de ce groupe ?

Les membres sont `grace`, `kevin` et `christian` (le quatrième champ dans `/etc/group`), mais seul `kevin` est l'administrateur du groupe (le troisième champ dans `/etc/gshadow`).

- Quel utilisateur ne peut pas se connecter au système ?

Le compte de l'utilisateur `henry` est verrouillé (il comporte un point d'exclamation devant son hash de mot de passe dans `/etc/shadow`).

- Quel utilisateur doit changer de mot de passe la prochaine fois qu'il se connectera au système ?

Si le troisième champ (Date du dernier changement de mot de passe) dans `/etc/shadow` est 0, l'utilisateur doit changer son mot de passe la prochaine fois qu'il se connectera au système. Par conséquent, `dave` doit changer son mot de passe.

- Combien de jours doivent s'écouler avant qu'un changement de mot de passe ne soit nécessaire pour `Carol` ?

60 jours (le cinquième champ dans `/etc/shadow`).

## Réponses aux Exercices d'Exploration

1. En tant que root, exécutez la commande `useradd -m dave` pour ajouter un nouveau compte utilisateur. Quelles sont les opérations effectuées par cette commande ? Supposons que `CREATE_HOME` et `USERGROUPS_ENAB` dans `/etc/login.defs` soient définis sur `yes`.

La commande ajoute un nouvel utilisateur nommé `dave`, à la liste des utilisateurs du système. Le répertoire personnel de `dave` est créé (par défaut `/home/dave`) et les fichiers et répertoires contenus dans le répertoire squelette sont copiés dans le répertoire personnel. Enfin, un nouveau groupe est créé avec le même nom que le compte utilisateur.

2. Maintenant que vous avez créé le compte `dave`, cet utilisateur peut-il se connecter au système ?

Non, parce que le compte `dave` est verrouillé (voir le point d'exclamation dans `/etc/shadow`).

```
# cat /etc/shadow | grep dave
dave:!:18015:0:99999:7:::
```

Si vous définissez un mot de passe pour `dave`, le compte sera déverrouillé. Vous pouvez le faire à l'aide de la commande `passwd`.

```
# passwd dave
Changing password for user dave.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

3. Identifier l'ID utilisateur (UID) et l'ID groupe (GID) de `dave` et de tous les membres du groupe `dave`.

```
# cat /etc/passwd | grep dave
dave:x:1015:1019::/home/dave:/bin/sh
# cat /etc/group | grep 1019
dave:x:1019:
```

L'UID et le GID de `dave` sont respectivement 1015 et 1019 (les troisième et quatrième champs dans `/etc/passwd`) et le groupe `dave` n'a pas de membres (le quatrième champ dans `/etc/group` est vide).

4. Créez les groupes `sys_admin`, `web_admin` et `db_admin` et identifiez leurs ID de groupe (GID).

```
# groupadd sys_admin
# groupadd web_admin
# groupadd db_admin
# cat /etc/group | grep admin
sys_admin:x:1020:
web_admin:x:1021:
db_admin:x:1022:
```

Les GID pour les groupes `sys_admin`, `web_admin` et `db_admin` sont respectivement 1020, 1021 et 1022.

- Ajoutez un nouveau compte utilisateur nommé `carol` avec l'UID 1035 et définir `sys_admin` comme son groupe primaire et `web_admin` et `db_admin` comme ses groupes secondaires.

```
# useradd -u 1035 -g 1020 -G web_admin,db_admin carol
# id carol
uid=1035(carol) gid=1020(sys_admin) groups=1020(sys_admin),1021(web_admin),1022(db_admin)
```

- Supprimez les comptes utilisateurs `dave` et `carol` et les groupes `sys_admin`, `web_admin` et `db_admin` que vous avez créés précédemment.

```
# userdel -r dave
# userdel -r carol
# groupdel sys_admin
# groupdel web_admin
# groupdel db_admin
```

- Exécutez la commande `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` et décrivez la sortie qu'elle vous donne en termes de permissions de fichiers. Lesquels de ces quatre fichiers sont ombragés (*shadowed*) pour des raisons de sécurité ? Supposons que votre système utilise des mots de passe "shadow".

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root 853 mag 1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag 1 08:00 /etc/gshadow
-rw-r--r-- 1 root root 1354 mag 1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag 1 08:00 /etc/shadow
```

Les fichiers `/etc/passwd` et `/etc/group` sont lisibles par tout le monde et sont filtrés pour des

raisons de sécurité. Lorsque les mots de passe shadow sont utilisés, vous pouvez voir un `x` dans le deuxième champ de ces fichiers car les mots de passe hachés des utilisateurs et des groupes sont stockés dans `/etc/shadow` et `/etc/gshadow`, qui ne sont lisibles que par root et, dans certains systèmes, également par les membres appartenant au groupe shadow.

8. Exécutez la commande `ls -l /usr/bin/passwd`. Quel bit spécial est défini et quelle est sa signification ?

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

La commande `passwd` a le bit SUID activé (le quatrième caractère de cette ligne), ce qui signifie que la commande est exécutée avec les privilèges du propriétaire du fichier (donc root). C'est ainsi que les utilisateurs ordinaires peuvent modifier leur mot de passe.



## 5.3 Gestion des propriétés et des droits d'accès aux fichiers

### Référence aux objectifs de LPI

[Linux Essentials version 1.6, Exam 010, Objective 5.3](#)

### Valeur

2

### Domaines de connaissance les plus importants

- Droits d'accès et propriété des fichiers et répertoires

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `ls -l`, `ls -a`
- `chmod`, `chown`



## 5.3 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	5 Sécurité et Permissions des Fichiers
<b>Objectif :</b>	5.3 Gestion des Permissions et de la Propriété des Fichiers
<b>Lesson:</b>	1 sur 1

### Introduction

Étant un système multi-utilisateurs, Linux a besoin d'un moyen de savoir à qui appartient chaque fichier et si un utilisateur est autorisé ou non à effectuer des actions sur ce fichier. Ceci afin de garantir la confidentialité des utilisateurs qui pourraient vouloir garder le contenu de leurs fichiers confidentiel, ainsi que pour assurer la collaboration en rendant certains fichiers accessibles à plusieurs utilisateurs.

Cela se fait grâce à un système de permissions à trois niveaux : chaque fichier sur le disque appartient à un utilisateur et à un groupe d'utilisateurs et possède trois ensembles de permissions : une pour son propriétaire, une pour le groupe qui possède le fichier et une pour tous les autres. Dans cette leçon, vous apprendrez comment interroger les permissions d'un fichier et comment les manipuler.

### Recherche d'Informations sur les Fichiers et les Répertoires

La commande `ls` est utilisée pour obtenir une liste du contenu de n'importe quel répertoire. Dans cette forme de base, vous obtenez uniquement les noms de fichiers :

```
$ ls
Another_Directory  picture.jpg  text.txt
```

Mais il y a beaucoup plus d'informations disponibles pour chaque fichier, y compris le type, la taille, la propriété et plus encore. Pour voir ces informations, vous devez demander à `ls` un listage en "forme longue", en utilisant le paramètre `-l` :

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol 4096 Dec 10 15:57 Another_Directory
-rw----- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol 1881 Dec 10 15:57 text.txt
```

Chaque colonne de la sortie ci-dessus a une signification :

- La *première* colonne de la liste indique le type de fichier et les permissions.

Par exemple, pour `drwxrwxr-x`:

- Le premier caractère, `d`, indique le type de fichier.
- Les trois caractères suivants, `rw`, indiquent les permissions pour le propriétaire du fichier, également appelé *user* ou `u`.
- Les trois caractères suivants, `rw`, indiquent les permissions du *groupe* propriétaire du fichier, également appelé `g`.
- Les trois derniers caractères, `r-x`, indiquent les permissions pour toute autre utilisateur, également appelée *others* ou `o`.
- La *deuxième* colonne indique le nombre de liens physiques pointant vers ce fichier. Pour un répertoire, cela signifie le nombre de sous-répertoires, plus un lien vers lui-même (`.`) et le répertoire parent (`..`).
- Les *troisième* et *quatrième* colonnes indiquent les informations relatives à la propriété : respectivement l'utilisateur et le groupe propriétaire du fichier.
- La *cinquième* colonne indique la taille des fichiers, en octets.
- La *sixième* colonne indique la date et l'heure précises, ou l'*horodatage*, de la dernière modification du fichier.
- La *septième* et dernière colonne indique le nom du fichier.

Si vous souhaitez voir les tailles de fichiers au format "lisible par l'homme", ajoutez le paramètre



-h à ls. Les fichiers de moins d'un kilooctet auront la taille indiquée en octets. Les fichiers de plus d'un kilooctet et de moins d'un mégaoctet auront un K ajouté après la taille, indiquant que la taille est en kilooctets. Il en va de même pour les fichiers dont la taille se situe dans les plages des mégaoctets (M) et des gigaoctets (G) :

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 Another_Directory
----r--r-- 1 carol carol  0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
-rw----- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol  33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

Pour n'afficher que les informations sur un ensemble spécifique de fichiers, ajoutez les noms de ces fichiers à ls :

```
$ ls -lh HugeFile.zip test.sh
total 1,2G
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
---xr-xr-x 1 carol carol  33 Dec 11 10:36 test.sh
```

## Et les Répertoires ?

Si vous essayez d'obtenir des informations sur un répertoire en utilisant `ls -l`, vous obtiendrez à la place la liste du contenu du répertoire :

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Pour éviter cela et avoir les informations sur le répertoire lui-même, ajoutez le paramètre `-d` à ls :

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

## Voir les Fichiers Cachés

La liste des répertoires que nous avons récupérée à l'aide de `ls -l` auparavant est incomplète :

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory
-rw----- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol 1881 Dec 10 15:57 text.txt
```

Il y a trois autres fichiers dans ce répertoire, mais ils sont cachés. Sous Linux, les fichiers dont le nom commence par un point (.) sont automatiquement cachés. Pour les voir, nous devons ajouter le paramètre `-a` à `ls` :

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol 4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol 4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory
-rw----- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol 1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol 0 Dec 10 16:01 .thisIsHidden
```

Le fichier `.thisIsHidden` est simplement caché parce que son nom commence par `.`.

Les répertoires `.` et `..` sont cependant spéciaux. `.` est un pointeur vers le répertoire courant, tandis que `..` est un pointeur vers le répertoire parent (le répertoire qui contient le répertoire courant). Sous Linux, chaque répertoire contient au moins ces deux répertoires spéciaux.

Vous pouvez combiner plusieurs paramètres pour `ls` (et de nombreuses autres commandes Linux). `ls -l -a` peut, par exemple, être écrit comme `ls -la`.

## Comprendre les Types de Fichiers

Nous avons déjà mentionné que la première lettre de chaque sortie de `ls -l` décrit le type de fichier. Les trois types de fichiers les plus courants sont les suivants :

### - (fichier normal)

Un fichier peut contenir des données de toute nature. Les fichiers peuvent être modifiés,

déplacés, copiés et supprimés.

### **d (répertoire)**

Un répertoire contient d'autres fichiers ou répertoires et aide à organiser le système de fichiers. Techniquement, les répertoires sont un type de fichier particulier.

### **l (lien souple)**

Ce "fichier" est un pointeur vers un autre fichier ou répertoire ailleurs dans le système de fichiers.

En plus de ceux-ci, il existe trois autres types de fichiers que vous devriez au moins connaître, mais qui ne sont pas couverts par cette leçon :

### **b (périphérique à blocs)**

Ce fichier représente un périphérique virtuel ou physique, généralement des disques ou d'autres types de périphérique de stockage. Par exemple, le premier disque dur du système peut être représenté par `/dev/sda`.

### **c (périphérique à caractères)**

Ce fichier représente un périphérique virtuel ou physique. Les terminaux (comme le terminal principal sur `/dev/ttyS0`) et les ports série sont des exemples courants de périphériques à caractères.

### **s (socket)**

Les sockets servent de "conduits" transmettant les informations entre deux programmes.

Ne modifiez aucune des permissions sur les périphériques à blocs, les périphériques à caractères ou les sockets, à moins que vous ne sachiez ce que vous faites. Cela pourrait empêcher votre système de fonctionner !

## **Comprendre les Permissions**

Dans la sortie de `ls -l`, les permissions de fichier sont affichées juste après le type de fichier, sous la forme de trois groupes de trois caractères chacun, dans l'ordre `r`, `w` et `x`. Voici ce qu'elles signifient. Gardez à l'esprit qu'un tiret `-` représente l'absence d'une permission particulière.

### **Permissions sur les Fichiers**

**r**

Signifie *read* (lecture) et a une valeur octale de 4 (ne vous inquiétez pas, nous discuterons bientôt des octales). Cela signifie qu'il est permis d'ouvrir un fichier et d'en lire le contenu.

**w**

Signifie *write* (écriture) et a une valeur octale de 2. Cela induit la permission de modifier ou de supprimer un fichier.

**x**

Signifie *execute* et a une valeur octale de 1, ce qui signifie que le fichier peut être exécuté en tant qu'exécutable ou script.

Ainsi, par exemple, un fichier avec les permissions `rw-` peut être lu et écrit, mais ne peut pas être exécuté.

## Permissions sur les Répertoires

**r**

Signifie *read* et a une valeur octale de 4, ce qui signifie qu'il est permis de lire le contenu du répertoire, comme les noms de fichiers. Mais cela n'implique *pas* la permission de lire les fichiers eux-mêmes.

**w**

Signifie *write* et a une valeur octale de 2. Il s'agit de la permission de créer ou de supprimer des fichiers dans un répertoire, ou de modifier leurs noms, permissions et propriétaires. Si un utilisateur a la permission d'écrire sur un répertoire, il peut changer les permissions de n'importe quel fichier dans le répertoire, même s'il n'a pas de permissions sur le fichier ou si le fichier appartient à un autre utilisateur.

**x**

Signifie *execute* et a une valeur octale de 1, Il s'agit d'une permission d'entrer dans un répertoire, mais pas de lister ses fichiers (pour cela, la permission `r` est nécessaire).

La dernière partie sur les répertoires peut sembler un peu confuse. Imaginons, par exemple, que vous ayez un répertoire nommé `Another_Directory`, avec les permissions suivantes :

```
$ ls -ld Another_Directory/
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Imaginez aussi que dans ce répertoire vous avez un script shell appelé `hello.sh` avec les

permissions suivantes :

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Si vous êtes l'utilisateur `carol` et que vous essayez de lister le contenu de `Another_Directory`, vous obtiendrez un message d'erreur, car votre utilisateur n'a pas les droits de lecture pour ce répertoire :

```
$ ls -l Another_Directory/  
ls: cannot open directory 'Another_Directory/': Permission denied
```

Cependant, l'utilisateur `carol` a les droits d'exécution, ce qui signifie qu'il peut entrer dans le répertoire. Par conséquent, l'utilisateur `carol` peut accéder aux fichiers à l'intérieur du répertoire, à condition qu'il ait les permissions correctes *pour le fichier en question*. Dans cet exemple, l'utilisateur a les permissions complètes pour le script `hello.sh`, ce qui lui permet d'exécuter le script, même s'il *ne peut pas* lire le contenu du répertoire qui le contient. On a uniquement besoin du nom complet du fichier.

```
$ sh Another_Directory/hello.sh  
Hello LPI World!
```

Comme nous l'avons dit précédemment, les permissions sont spécifiées dans l'ordre : d'abord pour le propriétaire du fichier, puis pour le groupe propriétaire, et enfin pour les autres utilisateurs. Chaque fois que quelqu'un tente d'effectuer une action sur le fichier, les permissions sont vérifiées de la même manière. Le système vérifie d'abord si l'utilisateur actuel est le propriétaire du fichier, et si c'est le cas, il applique uniquement le premier ensemble de permissions. Sinon, il vérifie si l'utilisateur actuel appartient au groupe propriétaire du fichier. Dans ce cas, il applique uniquement le deuxième ensemble de permissions. Dans tous les autres cas, le système appliquera le troisième ensemble des permissions. Cela signifie que si l'utilisateur actuel est le propriétaire du fichier, seules les permissions du propriétaire sont effectives, même si les permissions du groupe ou des autres sont plus permissives que celles du propriétaire.

## Modification des Permissions des Fichiers

La commande `chmod` est utilisée pour modifier les permissions d'un fichier, et prend au moins deux paramètres : le premier décrit les permissions à modifier, et le second indique le fichier ou le répertoire où la modification sera effectuée. Cependant, les permissions de modification peuvent être décrites de deux manières différentes, ou "modes".

Le premier, appelé *mode symbolique*, offre un contrôle fin, permettant d'ajouter ou de retirer une permission unique sans en modifier d'autres sur l'ensemble. L'autre mode, appelé *mode numérique*, est plus facile à mémoriser et plus rapide à utiliser si vous souhaitez définir toutes les valeurs de permission en même temps.

Les deux modes aboutiront au même résultat final. Ainsi, par exemple, les commandes :

```
$ chmod ug+rw-x,o-rwx text.txt
```

et

```
$ chmod 660 text.txt
```

produiront exactement le même résultat, un fichier avec les permissions fixées :

```
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Maintenant, comprenons comment chaque mode fonctionne.

## Mode Symbolique

Lorsque vous décrivez les permissions à modifier en *mode symbolique*, le(s) premier(s) caractère(s) indique(nt) les permissions que vous allez modifier : celles de l'utilisateur (u), du groupe (g), des autres (o) et/ou des trois ensemble (a).

Ensuite, vous devez dire à la commande ce qu'elle doit faire : vous pouvez accorder une permission (+), révoquer une permission (-) ou lui donner une valeur spécifique (=).

Enfin, vous précisez quelle permission vous souhaitez affecter : lecture (r), écriture (w) ou exécution (x).

Par exemple, imaginez que nous ayons un fichier appelé `text.txt` avec les permissions suivantes :

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Si vous souhaitez accorder des permissions d'écriture aux membres du groupe propriétaire du fichier, vous devez utiliser le paramètre `g+w`. C'est plus facile si vous y réfléchissez de cette

manière : “Pour le groupe (g), accorder (+) les droits d'écriture (w)”. Ainsi, la commande serait :

```
$ chmod g+w text.txt
```

Vérifions le résultat avec `ls` :

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Si vous voulez supprimer les permissions de lecture pour le propriétaire du même fichier, pensez ainsi : “Pour l'utilisateur (u) révoquer (-), les permissions de lecture (r)”. Le paramètre est donc `u-r`, comme ça :

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Et si nous voulons fixer les permissions exactement `rw-` à tout le monde ? Alors pensez ainsi : “Pour tous (a), fixer exactement (=), lecture (r), écriture (w), et non-exécution (-)”. Donc :

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Bien entendu, il est possible de modifier plusieurs permissions en même temps. Dans ce cas, séparez-les par une virgule (,) :

```
$ chmod u+rw,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

L'exemple ci-dessus peut être lu comme suit : “Pour l'utilisateur (u), accorder (+) des permissions de lecture, d'écriture et d'exécution (rwx), pour le groupe (g), révoquer (-), la permission d'exécution (x)”.

Lorsqu'elle est exécutée sur un répertoire, `chmod` ne modifie que les permissions du répertoire. `chmod` a un mode récursif, utile lorsque vous souhaitez modifier les permissions pour "tous les fichiers à l'intérieur d'un répertoire et de ses sous-répertoires". Pour l'utiliser, ajoutez le

paramètre `-R` après le nom de la commande et avant les permissions à modifier, comme ceci :

```
$ chmod -R u+rwX Another_Directory/
```

Cette commande peut être lue comme : "Récursivement (`-R`), pour l'utilisateur (`u`), accorder (`+`) les permissions de lecture, d'écriture et d'exécution (`rwX`)".

Soyez prudent et réfléchissez à deux fois avant d'utiliser l'option `-R`, car il est facile de changer les permissions sur des fichiers et des répertoires que vous ne voulez pas modifier, en particulier sur les répertoires contenant un grand nombre de fichiers et de sous-répertoires.

## Mode Numérique

En *mode numérique*, les permissions sont spécifiées d'une manière différente : sous la forme d'une valeur numérique à trois chiffres en notation octale, un système numérique en base 8.

Chaque permission a une valeur correspondante, et elles sont spécifiées dans l'ordre suivant : d'abord vient lecture (`r`), qui est 4, puis écriture (`w`), qui est 2 et enfin exécution (`x`), représentée par 1. S'il n'y a pas de permission, utilisez la valeur zéro (`0`). Ainsi, une permission `rwX` serait 7 ( $4+2+1$ ) et `r-x` serait 5 ( $4+0+1$ ).

Le premier des trois chiffres de l'ensemble des permissions représente les permissions pour l'utilisateur (`u`), le deuxième pour le groupe (`g`) et le troisième pour les autres (`o`). Si nous voulions fixer les permissions pour un fichier à `rw-rw----`, la valeur octale serait de `660` :

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

En outre, la syntaxe en *mode numérique* est la même qu'en *mode symbolique*, le premier paramètre représente les permissions que vous souhaitez modifier, et le second pointe vers le fichier ou le répertoire où la modification sera effectuée.

Si la valeur d'une permission est *étrange*, le fichier est sûrement exécutable !

Quelle syntaxe utiliser ? Le *mode numérique* est recommandé si vous souhaitez modifier les



permissions pour une valeur spécifique, par exemple 640 (rw- r-- ---).

Le *mode symbolique* est plus utile si vous voulez permuter une valeur spécifique, indépendamment des permissions actuelles du fichier. Par exemple, je peux ajouter des permissions d'exécution pour l'utilisateur en utilisant simplement `chmod u+x script.sh` sans tenir compte, ni même toucher, aux permissions actuelles pour le groupe et les autres.

## Modification de la Propriété des Fichiers

La commande `chown` est utilisée pour modifier la propriété d'un fichier ou d'un répertoire. La syntaxe est assez simple :

```
chown utilisateur:groupe fichiercible
```

Par exemple, vérifions un fichier appelé `text.txt` :

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

L'utilisateur qui possède le fichier est `carol`, et le groupe est également `carol`. Maintenant, modifions le groupe propriétaire du fichier pour un autre groupe, comme `students` :

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Gardez à l'esprit que l'utilisateur qui possède un fichier n'a pas besoin d'appartenir au groupe qui possède le fichier. Dans l'exemple ci-dessus, il n'est pas nécessaire que l'utilisateur soit membre du groupe `students`. Toutefois, il doit être membre du groupe pour pouvoir transférer la propriété du fichier à ce groupe.

L'utilisateur ou le groupe peut être omis si vous ne souhaitez pas les modifier. Ainsi, pour changer uniquement le groupe possédant un fichier, vous utiliserez `chown :students text.txt`. Pour changer uniquement l'utilisateur, la commande sera `chown carol text.txt`. Vous pouvez aussi utiliser la commande `chgrp students text.txt` pour ne changer que le groupe.

Sauf si vous êtes l'administrateur du système (root), vous ne pouvez pas changer la propriété d'un fichier appartenant à un autre utilisateur ou à un groupe auquel vous n'appartenez pas. Si vous essayez de le faire, vous obtiendrez le message d'erreur `Opération non autorisée`.

## Interroger les Groupes

Avant de changer la propriété d'un fichier, il peut être utile de savoir quels groupes existent sur le système, quels utilisateurs sont membres d'un groupe et à quels groupes appartient un utilisateur. Ces tâches peuvent être accomplies à l'aide de deux commandes, `groups` et `groupmems`.

Pour voir quels groupes existent sur votre système, il suffit de taper `groups` :

```
$ groups
carol students cdrom sudo dip plugdev lpadmin sambashare
```

Et si vous voulez savoir à quels groupes appartient un utilisateur, ajoutez le nom d'utilisateur comme paramètre :

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Pour faire l'inverse, en affichant quels utilisateurs appartiennent à un groupe, utilisez `groupmems`. Le paramètre `-g` spécifie le groupe, et `-l` liste tous ses membres :

```
$ sudo groupmems -g cdrom -l
carol
```

`groupmems` ne peut être exécuté qu'en tant que root, l'administrateur du système. Si vous n'êtes pas actuellement connecté en tant que root, ajoutez `sudo` avant la commande.

## Permissions Spéciales

Outre les permissions de lecture, d'écriture et d'exécution pour les utilisateurs, les groupes et les autres, chaque fichier peut avoir trois autres *permissions spéciales* qui peuvent modifier le fonctionnement d'un répertoire ou l'exécution d'un programme. Elles peuvent être spécifiées en mode symbolique ou numérique, et sont les suivantes :

### Le Sticky Bit

Le sticky bit (bit collant), également appelé *étiquette de suppression restreinte*, a la valeur octale 1 et, en mode symbolique, est représenté par un `t` dans les permissions des autres. Cela ne

s'applique qu'aux répertoires et, sous Linux, cela empêche les utilisateurs de supprimer ou de renommer un fichier dans un répertoire, à moins qu'ils ne soient propriétaires de ce fichier ou de ce répertoire.

Les répertoires avec le sticky bit activé montrent un **t** remplaçant le **x** sur les permissions des *autres* sur la sortie de `ls -l` :

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

En mode numérique, les permissions spéciales sont spécifiées à l'aide d'une "notation à 4 chiffres", le premier chiffre représentant la permission spéciale sur laquelle agir. Par exemple, pour définir le sticky bit (valeur 1) pour le répertoire `Another_Directory` en mode numérique, avec les permissions 755, la commande serait :

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

## Le Set GID

Le Set GID, également connu sous le nom SGID ou bit Set Group ID, a la valeur octale 2 et en mode symbolique est représenté par un **s** sur les permissions du *groupe*. Cela peut s'appliquer aux fichiers ou répertoires exécutables. Sur les fichiers exécutables, il accordera au processus résultant de l'exécution du fichier l'accès aux privilèges du groupe qui possède le fichier. Lorsqu'il est appliqué aux répertoires, il fera en sorte que chaque fichier ou répertoire créé sous lui hérite du groupe du répertoire parent.

Les fichiers et répertoires avec un bit SGID montrent un **s** remplaçant le **x** sur les permissions du *groupe* sur la sortie de `ls -l` :

```
$ ls -l test.sh
-rwxr-sr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Pour ajouter des permissions SGID à un fichier en mode symbolique, la commande serait

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root      33 Dec 11 10:36 test.sh
```

L'exemple suivant vous permettra de mieux comprendre les effets du SGID sur un répertoire. Supposons que nous ayons un répertoire appelé `Sample_Directory`, appartenant à l'utilisateur `carol` et au groupe `users`, avec la structure de permission suivante :

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Maintenant, entrons dans ce répertoire, et à l'aide de la commande `touch` créons un fichier vide à l'intérieur. Le résultat serait :

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Comme on peut le voir, le fichier est la propriété de l'utilisateur `carol` et du groupe `carol`. Mais, si le répertoire avait les permissions SGID définies, le résultat serait différent. Tout d'abord, ajoutons le bit SGID au répertoire `Sample_Directory` et vérifions les résultats :

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

Le `s` sur les permissions de groupe indique que le bit SGID est activé. Maintenant, rentrons dans ce répertoire et, à nouveau, créons un fichier vide avec la commande `touch` :

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
-rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

Comme on peut le voir, le groupe qui possède le fichier est `users`. Cela est dû au fait que le bit SGID a fait que le fichier hérite du groupe propriétaire de son répertoire parent, qui est `users`.

## Le Set UID

Le SUID, également connu sous le nom de Set User ID, a la valeur octale `4` et est représenté par un `s` sur les permissions de l'utilisateur en mode symbolique. Il s'applique uniquement aux fichiers et son comportement est similaire au bit SGID, mais le processus s'exécutera avec les privilèges de

l'*utilisateur* qui possède le fichier. Les fichiers avec le bit SUID montrent un `s` remplaçant le `x` sur les permissions de l'utilisateur sur la sortie de `ls -l` :

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Vous pouvez combiner plusieurs permissions spéciales dans un même paramètre en les additionnant. Ainsi, pour mettre SGID (valeur 2) et SUID (valeur 4) en mode numérique pour le script `test.sh` avec les permissions 755, vous devez taper :

```
$ chmod 6755 test.sh
```

Et le résultat serait :

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

Si votre terminal prend en charge la couleur, ce qui est le cas de la plupart des terminaux actuels, vous pouvez rapidement voir si ces permissions spéciales sont définies en jetant un coup d'œil à la sortie de `ls -l`. Pour le stick bit, le nom du répertoire peut être affiché dans une police noire sur fond bleu. Il en va de même pour les fichiers avec les bits SGID (fond jaune) et SUID (fond rouge). Les couleurs peuvent être différentes selon la distribution Linux et les paramètres du terminal que vous utilisez.

## Exercices Guidés

1. Créez un répertoire nommé `emptydir` en utilisant la commande `mkdir emptydir`. Maintenant, à l'aide de `ls`, listez les permissions du répertoire `emptydir`.

2. Créez un fichier vide nommé `emptyfile` avec la commande `touch emptyfile`. Maintenant, en utilisant `chmod` avec la notation symbolique, ajoutez les permissions d'exécution pour le propriétaire du fichier `emptyfile`, et supprimez les permissions d'écriture et d'exécution pour tous les autres. Faites cela en utilisant une seule commande `chmod`.

3. Quelles seront les permissions pour un fichier appelé `text.txt` après que vous aurez utilisé la commande `chmod 754 text.txt` ?

4. Supposons qu'un fichier nommé `test.sh` soit un script shell avec les permissions et la propriété suivantes :

```
-rwxr-sr-x 1 carol root    33 Dec 11 10:36 test.sh
```

- Quelles sont les permissions pour le propriétaire du fichier ?
- Si l'utilisateur `john` exécute ce script, sous quels privilèges d'utilisateur sera-t-il exécuté ?
- En utilisant la notation numérique, que devrait être la syntaxe de `chmod` pour "annuler" la permission spéciale accordée à ce fichier ?

5. Examinez ce fichier :

```
$ ls -l /dev/sdb1  
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

Quel type de fichier est `sdb1` ? Et qui peut y écrire ?

## 6. Examinez les 4 fichiers suivants :

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Notez les permissions correspondantes pour chaque fichier et répertoire en utilisant la notation numérique à 4 chiffres.

### **Another\_Directory**

### **foo.bar**

### **HugeFile.zip**

### **Sample\_Directory**

## Exercices d'Exploration

1. Essayez ceci sur un terminal : créez un fichier vide appelé `emptyfile` avec la commande `touch emptyfile`. Maintenant, mettez à zéro les permissions pour le fichier avec `chmod 000 emptyfile`. Que se passe-t-il si vous modifiez les permissions pour `emptyfile` en passant *une* seule valeur pour `chmod` en mode numérique, par exemple `chmod 4 emptyfile` ? Et si nous en utilisons deux, comme `chmod 44 emptyfile` ? Que pouvons-nous apprendre sur la façon dont `chmod` lit la valeur numérique ?

---

2. Pouvez-vous exécuter un fichier pour lequel vous avez des permissions d'exécution, mais pas de lecture (`--x`) ? Pourquoi ou pourquoi pas ?

---

3. Considérez les permissions pour le répertoire temporaire sur un système Linux, `/tmp` :

```
$ ls -l /tmp
drwxrwxrwt 19 root root 16K Dec 21 18:58 tmp
```

L'utilisateur, le groupe et les autres ont toutes les permissions. Mais un utilisateur régulier peut-il supprimer *des* fichiers dans ce répertoire ? Pourquoi ?

---

4. Un fichier nommé `test.sh` dispose des permissions suivantes : `-rwsr-xr-x`, ce qui signifie que le bit SUID est activé. Maintenant, exécutez les commandes suivantes :

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Qu'avons-nous fait ? Que signifie le `S` majuscule ?

---

5. Comment créer un répertoire nommé `Box` dans lequel tous les fichiers sont automatiquement détenus par le groupe `users` et ne peuvent être supprimés que par l'utilisateur qui les a créés ?

---



## Résumé

En tant que système multi-utilisateurs, Linux a besoin d'un moyen de savoir qui possède chaque fichier et qui peut y accéder. Cela se fait par un système de permissions à trois niveaux, et dans cette leçon nous avons appris tout sur le fonctionnement de ce système.

Dans cette leçon, vous avez appris comment utiliser `ls` pour obtenir des informations sur les permissions de fichiers, comment contrôler ou changer qui peut créer, supprimer ou modifier un fichier avec `chmod`, à la fois en notation *numérique* et *symbolique* et comment changer la propriété des fichiers avec `chown` et `chgrp`.

Les commandes suivantes ont été abordées dans cette leçon :

### `ls`

Liste les fichiers, en incluant éventuellement des détails tels que les permissions.

### `chmod`

Modifie les permissions d'un fichier ou d'un répertoire.

### `chown`

Change l'utilisateur et/ou le groupe propriétaire d'un fichier ou d'un répertoire.

### `chgrp`

Change le groupe propriétaire d'un fichier ou d'un répertoire.

## Réponses aux Exercices Guidés

1. Créez un répertoire nommé `emptydir` en utilisant la commande `mkdir emptydir`. Maintenant, à l'aide de `ls`, listez les permissions du répertoire `emptydir`.

Ajoutez le paramètre `-d` à `ls` pour voir les attributs du répertoire, au lieu de lister son contenu. La réponse est donc :

```
ls -l -d emptydir
```

Points bonus si vous avez fusionné les deux paramètres en un seul, comme dans `ls -ld emptydir`.

2. Créez un fichier vide nommé `emptyfile` avec la commande `touch emptyfile`. Maintenant, en utilisant `chmod` en notation symbolique, ajoutez les permissions d'exécution pour le propriétaire du fichier `emptyfile`, et supprimez les permissions d'écriture et d'exécution pour tous les autres. Faites cela en utilisant une seule commande `chmod`.

Pensez-y de cette façon :

- “Pour l'utilisateur qui possède le fichier (u) ajouter (+) la permission d'exécution (x)”, donc `u+x`.
- “Pour le groupe (g) et les autres utilisateurs (o), supprimer (-) les permissions d'écriture (w) et d'exécution (x)”, donc `go-wx`.

Pour combiner ces deux séries de permissions, nous ajoutons une virgule entre elles. Le résultat final est donc le suivant :

```
chmod u+x,go-wx emptyfile
```

3. Quelles seront les permissions d'un fichier appelé `text.txt` après que j'aurai utilisé la commande `chmod 754 text.txt` ?

Rappelez-vous qu'en notation numérique, chaque chiffre représente un ensemble de trois permissions, chacune avec une valeur respective : *lecture* est 4, *écriture* est 2, *exécution* est 1 et aucune permission est 0. Nous obtenons la valeur d'un chiffre en additionnant les valeurs correspondantes pour chaque permission. 7 est 4+2+1, ou `rwX`, 5 est 4+0+1, donc `r-x` et 4 est juste lecture, ou `r--`. Les permissions pour `text.txt` seraient :

```
rwXr-xr--
```

4. Supposons qu'un fichier nommé `test.sh` soit un script shell avec les permissions et la propriété suivantes :

```
-rwxr-sr-x 1 carol root    33 Dec 11 10:36 test.sh
```

- Quelles sont les permissions pour le propriétaire du fichier ?

Les permissions pour le propriétaire (du 2nd au 4ième caractères dans la sortie de `ls -l`) sont `rwX`, donc la réponse est : "de lire, d'écrire et d'exécuter le fichier".

- Si l'utilisateur `john` exécute ce script, sous quels privilèges d'utilisateur sera-t-il exécuté ?

Faites attention aux permissions pour le *groupe*. Elles sont `r-s`, ce qui signifie que le bit SGID est activé. Le groupe qui possède ce fichier est `root`, donc le script, même s'il est lancé par un utilisateur normal, sera exécuté avec les privilèges de l'utilisateur.

- En utilisant la notation numérique, que devrait être la syntaxe de `chmod` pour "annuler" la permission spéciale accordée à ce fichier ?

Nous pouvons "annuler" les permissions spéciales en passant un quatrième chiffre, `0`, à `chmod`. Les permissions actuelles sont `755`, donc la commande devrait être `chmod 0755`.

5. Examinez ce fichier :

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

Quel type de fichier est `sdb1` ? Et qui peut y écrire ?

Le premier caractère de la sortie de `ls -l` indique le type de fichier. `b` est un *périphérique à bloc*, généralement un disque (interne ou externe), connecté à la machine. Le propriétaire (`root`) et tous les utilisateurs du groupe `disk` peuvent y écrire.

6. Examinez les 4 fichiers suivants :

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
```

```
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Notez les permissions correspondantes pour chaque fichier et répertoire en utilisant la notation numérique à 4 chiffres.

Les permissions correspondantes, en notation numérique, sont les suivantes :

### **Another\_Directory**

Réponse : 1755

1 pour le sticky bit, 755 pour les permissions régulières (rwx pour l'utilisateur, r-x pour le groupe et les autres).

### **foo.bar**

Réponse : 0044

Pas de permissions spéciales (donc le premier chiffre est 0), pas de permissions pour l'utilisateur (---) et juste lire (r--r--) pour le groupe et les autres.

### **HugeFile.zip**

Réponse : 0664

Aucune permission spéciale, donc le premier chiffre est 0. 6 (rw-) pour l'utilisateur et le groupe, 4 (r--) pour les autres.

### **Sample\_Directory**

Réponse : 2755

2 pour le bit SGID, 7 (rwx) pour l'utilisateur, 5 (r-x) pour le groupe et les autres.

## Réponses aux Exercices d'Exploration

1. Essayez ceci sur un terminal : créez un fichier vide appelé `emptyfile` avec la commande `touch emptyfile`. Maintenant, mettez à zéro les permissions pour le fichier avec `chmod 000 emptyfile`. Que se passe-t-il si vous modifiez les permissions pour `emptyfile` en passant *une* seule valeur pour `chmod` en mode numérique, par exemple `chmod 4 emptyfile` ? Et si nous en utilisons deux, comme `chmod 44 emptyfile` ? Que pouvons-nous apprendre sur la façon dont `chmod` lit la valeur numérique ?

N'oubliez pas que nous avons “mis à zéro” les permissions de `emptyfile`. Ainsi, son état initial serait :

```
----- 1 carol carol 0 Dec 11 10:55 emptyfile
```

Maintenant, essayons la première commande, `chmod 4 emptyfile` :

```
$ chmod 4 emptyfile
$ ls -l emptyfile
-----r-- 1 carol carol 0 Dec 11 10:55 emptyfile
```

Les permissions pour les *autres* ont été modifiées. Et si nous essayons deux chiffres, comme `chmod 44 emptyfile` ?

```
$ chmod 44 emptyfile
$ ls -l emptyfile
----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
```

Maintenant, les permissions pour le *groupe* et les *autres* ont été affectées. On peut en conclure qu'en notation numérique, `chmod` lit la valeur "à l'envers", du chiffre le moins significatif (*autres*) au plus significatif (*utilisateur*). Si vous passez un chiffre, vous modifiez les permissions pour les *autres*. Avec deux chiffres, vous modifiez le *groupe* et les *autres*, et avec trois chiffres, vous modifiez l'*utilisateur*, le *groupe* et les *autres* et avec quatre chiffres, vous modifiez l'*utilisateur*, le *groupe*, les *autres* et les permissions spéciales.

2. Pouvez-vous exécuter un fichier pour lequel vous avez des permissions d'exécution, mais pas de lecture (`--x`) ? Pourquoi ou pourquoi pas ?

Au premier abord, la réponse semble évidente : si vous avez la permission d'exécuter, le fichier devrait s'exécuter. Cependant, sans permission de lecture, le système ne peut pas ouvrir le

fichier et lire son contenu pour l'exécuter. Par conséquent, sans permission de lecture, vous ne pouvez pas exécuter un fichier, même si vous avez une permission d'exécution pour celui-ci.

3. Considérez les permissions pour le répertoire temporaire sur un système Linux, `/tmp` :

```
$ ls -l /tmp
drwxrwxrwt 19 root root 16K Dec 21 18:58 tmp
```

L'utilisateur, le groupe et les autres ont toutes les permissions. Mais un utilisateur régulier peut-il supprimer *des* fichiers dans ce répertoire ? Pourquoi ?

`/tmp` est ce que nous appelons un répertoire *accessible en écriture pour tous*, ce qui signifie que tout utilisateur peut y écrire. Mais nous ne voulons pas qu'un utilisateur modifie des fichiers créés par d'autres, donc le *sticky bit* est activé (comme indiqué par le `t` sur les permissions pour les *autres*). Cela signifie qu'un utilisateur peut supprimer des fichiers dans `/tmp`, mais seulement s'il a créé ces fichiers.

4. Un fichier appelé `test.sh` dispose des permissions suivantes : `-rwsr-xr-x`, ce qui signifie que le bit SUID est activé. Maintenant, exécutez les commandes suivantes :

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Qu'avons-nous fait ? Que signifie le `S` majuscule ?

Nous avons supprimé les permissions d'exécution pour l'utilisateur qui possède le fichier. Le `s` (ou `t`) prend la place du `x` sur la sortie de `ls -l`, le système a donc besoin d'un moyen de montrer si l'utilisateur a des permissions d'exécution ou non. Il le fait en modifiant la casse du caractère spécial.

Un `s` minuscule sur le premier groupe de permissions signifie que l'utilisateur qui possède le fichier a des permissions d'exécution et que le bit SUID est activé. Un `S` majuscule signifie que l'utilisateur qui possède le fichier n'a pas (-) les permissions d'exécution et que le bit SUID est activé.

On peut en dire autant du SGID. Un `s` minuscule sur le deuxième groupe de permissions signifie que le groupe qui possède le fichier a des permissions d'exécution et que le bit SGID est activé. Un `S` majuscule signifie que le groupe qui possède le fichier n'a pas les permissions d'exécution (-) et que le bit SGID est activé.

C'est également vrai pour le sticky bit, représenté par le `t` dans le troisième groupe des permissions. Un `t` minuscule signifie que le sticky bit est activé et que les autres ont des permissions d'exécution. `T` majuscule signifie que le sticky bit est activé et que les autres n'ont pas de droits d'exécution.

5. Comment créer un répertoire nommé `Box` dans lequel tous les fichiers sont automatiquement détenus par le groupe `users` et ne peuvent être supprimés que par l'utilisateur qui les a créés ?

Il s'agit d'un processus en plusieurs étapes. La première étape consiste à créer le répertoire :

```
$ mkdir Box
```

Nous voulons que chaque fichier créé dans ce répertoire soit automatiquement attribué au groupe `users`. Nous pouvons le faire en définissant ce groupe comme propriétaire du répertoire, puis en y plaçant le bit SGID. Nous devons également nous assurer que tout membre du groupe peut écrire dans ce répertoire.

Comme nous ne nous soucions pas des autres permissions et que nous ne voulons "permuter" que les bits spéciaux, il est logique d'utiliser le mode symbolique :

```
$ chown :users Box/  
$ chmod g+wx Box/
```

Notez que si votre utilisateur actuel n'appartient pas au groupe `users`, vous devrez utiliser la commande `sudo` avant les commandes ci-dessus pour effectuer le changement en tant que root.

Maintenant, pour la dernière partie, il faut s'assurer que seul l'utilisateur qui a créé un fichier est autorisé à le supprimer. Cela se fait en plaçant le sticky bit (représenté par un `t`) sur le répertoire. N'oubliez pas qu'il est défini sur les permissions des autres (`o`).

```
$ chmod o+t Box/
```

Les permissions sur le répertoire `Box` doivent apparaître comme suit :

```
drwxrwsr-t 2 carol users 4,0K Jan 18 19:09 Box
```

Bien sûr, vous pouvez spécifier le SGID et le sticky bit en utilisant une seule commande `chmod` :

```
$ chmod g+ws,o+t Box/
```

Des points en bonus si vous y avez pensé.





## 5.4 Répertoires et fichiers spéciaux

### Référence aux objectifs de LPI

Linux Essentials v1.6, Exam 010, Objective 5.4

### Valeur

1

### Domaines de connaissance les plus importants

- Utilisation de fichiers et répertoires temporaires
- Liens symboliques

### Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif

- `/tmp/`, `/var/tmp/` et Sticky Bit
- `ls -d`
- `ln -s`



Linux  
Professional  
Institute

## 5.4 Leçon 1

<b>Certification :</b>	Linux Essentials
<b>Version :</b>	1.6
<b>Thème :</b>	5 Sécurité et Permissions de Fichiers
<b>Objectif :</b>	5.4 Répertoires et Fichiers Spéciaux
<b>Leçon:</b>	1 sur 1

### Introduction

Sous Linux, tout est traité comme un fichier. Cependant, certains fichiers reçoivent un traitement spécial, soit en raison de l'endroit où ils sont stockés, comme les fichiers temporaires, soit en raison de la manière dont ils interagissent avec le système de fichiers, comme les liens. Dans cette leçon, nous apprendrons où se trouvent ces fichiers, comment ils fonctionnent et comment les gérer.

### Fichiers Temporaires

Les fichiers temporaires sont des fichiers utilisés par les programmes pour stocker des données qui ne sont nécessaires que pour une courte durée. Il peut s'agir de données de processus en cours d'exécution, de journaux de crash, de fichiers de travail d'une sauvegarde automatique, de fichiers intermédiaires utilisés lors d'une conversion de fichier, de fichiers de cache, etc.

### Localisation des Fichiers Temporaires

La version 3.0 du *FHS* (Filesystem Hierarchy Standard) définit des emplacements standard pour les fichiers temporaires sur les systèmes Linux. Chaque emplacement a un but et un

comportement différents, il est recommandé aux développeurs de suivre les conventions établies par le FHS lorsqu'ils écrivent des données temporaires sur le disque.

### **/tmp**

Selon le FHS, les programmes ne doivent pas supposer que les fichiers écrits ici seront conservés entre les exécutions d'un programme. La recommandation est que ce répertoire soit effacé (tous les fichiers sont effacés) lors du démarrage du système, bien que cela *ne soit pas obligatoire*.

### **/var/tmp**

Un autre emplacement pour les fichiers temporaires, mais celui-ci *ne doit pas être effacé* pendant le démarrage du système, c'est-à-dire que les fichiers stockés ici persisteront généralement entre les redémarrages.

### **/run**

Ce répertoire contient les données des variables d'exécution utilisées par les processus en cours, telles que les fichiers d'identification des processus (.pid). Les programmes qui ont besoin de plus d'un fichier d'exécution peuvent créer des sous-répertoires ici. Cet emplacement *doit être effacé* lors du démarrage du système. Le but de ce répertoire était autrefois servi par /var/run, et sur certains systèmes, /var/run peut être un lien symbolique vers /run.

Notez que rien n'empêche un programme de créer des fichiers temporaires ailleurs dans le système, mais il est de bonne pratique de respecter les conventions fixées par le FHS.

## **Permissions sur les Fichiers Temporaires**

Le fait d'avoir des répertoires temporaires à l'échelle du système sur un système multi-utilisateurs présente quelques difficultés concernant les permissions d'accès. Au premier abord, on pourrait penser que de tels répertoires seraient accessibles en écriture à tous, c'est-à-dire que tout utilisateur pourrait y écrire ou supprimer des données. Mais si cela était vrai, comment pourrions-nous empêcher un utilisateur d'effacer ou de modifier des fichiers créés par un autre ?

La solution est une permission spéciale appelée *sticky bit* (bit collant), qui s'applique à la fois aux répertoires et aux fichiers. Cependant, pour des raisons de sécurité, le noyau Linux ignore le sticky bit lorsqu'il est appliqué aux fichiers. Lorsque ce bit spécial est défini pour un répertoire, il empêche les utilisateurs de supprimer ou de renommer un fichier dans ce répertoire à moins qu'ils ne soient propriétaires du fichier.

Les répertoires avec le sticky bit activé affichent un **t** remplaçant le **x** sur la permission des *autres* dans la sortie de `ls -l`. Par exemple, vérifions les permissions pour les répertoires /tmp et /var/tmp :

```
$ ls -ldh /tmp/ /var/tmp/
drwxrwxrwt 25 root root 4,0K Jun  7 18:52 /tmp/
drwxrwxrwt 16 root root 4,0K Jun  7 09:15 /var/tmp/
```

Comme vous pouvez le voir par le `t` remplaçant le `x` sur la permission des *autres*, les deux répertoires ont le sticky bit activé.

Pour activer le sticky bit sur un répertoire avec `chmod` en mode numérique, utilisez la notation à quatre chiffres avec 1 comme premier chiffre. Par exemple :

```
$ chmod 1755 temp
```

définira le sticky bit pour le répertoire nommé `temp` et les permissions comme `rwxr-xr-t`.

Lorsque vous utilisez le mode symbolique, utilisez le paramètre `t`. Donc, `+t` pour activer le sticky bit, et `-t` pour le désactiver. Ainsi :

```
$ chmod +t temp
```

## Comprendre les Liens

Nous avons déjà dit que sur Linux, tout est traité comme un fichier. Mais il existe un type de fichier *spécial*, appelé *lien*, et il y a deux types de liens sur un système Linux :

### Liens symboliques

Appelés aussi *liens souples*, ils pointent vers le chemin d'un autre fichier. Si vous supprimez le fichier vers lequel le lien pointe (appelé *cible*), le lien existera toujours, mais il "cessera de fonctionner", car il pointe maintenant vers "rien".

### Liens solides

Pensez à un lien solide comme un deuxième nom pour le fichier original. Ce *ne* sont *pas* des doublons, mais plutôt une entrée supplémentaire dans le système de fichiers pointant vers le même endroit (inode) sur le disque.

Un *inode* est une structure de données qui stocke les attributs d'un objet (comme un fichier ou un répertoire) sur un système de fichiers. Parmi ces attributs figurent le nom du fichier, les permissions, la propriété et les blocs du disque sur lesquels les données de l'objet sont stockées. Considérez cela comme une entrée dans un index, d'où le nom, qui vient de "index".

node”.

## Travailler avec des Liens Solides

### Création de Liens Solides

La commande pour créer un lien solide sur Linux est `ln`. La syntaxe de base est :

```
$ ln CIBLE NOM_DU_LIEN
```

La `CIBLE` doit déjà exister (c'est le fichier vers lequel le lien pointera), et si la cible ne se trouve pas dans le répertoire actuel, ou si vous voulez créer le lien ailleurs, vous *devez* indiquer le chemin complet vers celui-ci. Par exemple, la commande :

```
$ ln target.txt /home/carol/Documents/hardlink
```

va créer un fichier nommé `hardlink` dans le répertoire `/home/carol/Documents/`, lié au fichier `target.txt` du répertoire courant.

Si vous omettez le dernier paramètre (`NOM_DU_LIEN`), un lien portant le même nom que la cible sera créé dans le répertoire courant.

### Gestion des Liens Solides

Les liens solides sont des entrées dans le système de fichiers qui ont des noms différents mais qui pointent vers les mêmes données sur le disque. Tous ces noms sont identiques et peuvent être utilisés pour faire référence à un fichier. Si vous modifiez le contenu de l'un des noms, le contenu de tous les autres noms pointant vers ce fichier change puisque tous ces noms pointent vers les mêmes données. Si vous supprimez l'un des noms, les autres noms continueront à fonctionner.

Cela se produit parce que lorsque vous "supprimez" un fichier, les données ne sont pas réellement effacées du disque. Le système supprime simplement l'entrée de la table du système de fichiers qui pointe sur l'inode correspondant aux données du disque. Mais si vous avez une deuxième entrée pointant vers le même inode, vous pouvez toujours accéder aux données. Imaginez que deux routes convergent vers le même point. Même si vous bloquez ou redirigez l'une des routes, vous pouvez toujours atteindre la destination en utilisant l'autre.

Vous pouvez vérifier cela en utilisant le paramètre `-i` de `ls`. Considérez le contenu suivant d'un répertoire :

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

Le numéro précédant les permissions est le numéro d'inode. Vous voyez que le fichier `hardlink` et le fichier `target.txt` ont le même numéro (3806696) ? C'est parce que l'un est un lien solide de l'autre.

Mais lequel est l'original et lequel est le lien ? On ne peut pas vraiment le dire, car à toutes fins pratiques, ils sont identiques.

Notez que chaque lien solide pointant vers un fichier augmente le *nombre de liens* du fichier. C'est le nombre juste après les permissions sur la sortie de `ls -l`. Par défaut, chaque fichier a un nombre de liens de 1 (les répertoires ont un nombre de 2), et chaque lien solide vers ce fichier augmente le nombre de liens de un. C'est donc la raison pour laquelle le nombre de liens est 2 pour les fichiers de la liste ci-dessus.

Contrairement aux liens symboliques, vous ne pouvez créer que des liens solides vers des fichiers, et le lien et la cible doivent tous deux résider dans le même système de fichiers.

## Déplacement et Suppression des Liens Solides

Comme les liens solides sont traités comme des fichiers ordinaires, ils peuvent être supprimés avec `rm` et renommés ou déplacés dans le système de fichiers avec `mv`. Et puisqu'un lien solide pointe vers le même inode de la cible, il peut être déplacé librement, sans crainte de "casser" le lien.

## Liens Symboliques

### Création de Liens Symboliques

La commande utilisée pour créer un lien symbolique est également `ln`, mais avec le paramètre `-s` ajouté. Ainsi :

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Cela créera un fichier nommé `softlink` dans le répertoire `/home/carol/Documents/`, pointant vers le fichier `target.txt` dans le répertoire courant.

Comme pour les liens solides, vous pouvez omettre le nom du lien pour créer un lien portant le

même nom que la cible dans le répertoire courant.

## Gestion des Liens Symboliques

Les liens symboliques pointent vers un autre chemin dans le système de fichiers. Vous pouvez créer des liens symboliques vers des fichiers *et* des répertoires, même sur des partitions différentes. Il est assez facile de repérer un lien symbolique sur la sortie de `ls` :

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol  12 Jun  7 10:14 softlink -> target.txt
```

Dans l'exemple ci-dessus, le premier caractère sur les permissions du fichier `softlink` est `l`, indiquant un lien symbolique. En outre, juste après le nom du fichier, nous voyons le nom de la cible vers laquelle le lien pointe, le fichier `target.txt`.

Notez que sur les listes de fichiers et de répertoires, les liens symboliques eux-mêmes indiquent toujours les permissions `rwX` pour l'utilisateur, le groupe et les autres, mais en pratique, leurs permissions d'accès sont les mêmes que celles de la cible.

## Déplacement et Suppression des Liens Symboliques

Comme les liens solides, les liens symboliques peuvent être supprimés à l'aide de `rm` et déplacés ou renommés à l'aide de `mv`. Cependant, il faut faire particulièrement attention lors de leur création, pour éviter de “casser” le lien s'il est déplacé de son emplacement d'origine.

Lorsque vous créez des liens symboliques, vous devez être conscient que, à moins qu'un chemin ne soit entièrement spécifié, l'emplacement de la cible est interprété comme étant *relatif* à l'emplacement du lien. Cela peut créer des problèmes si le lien, ou le fichier vers lequel il pointe, est déplacé.

C'est plus facile à comprendre avec un exemple. Supposons que nous ayons un fichier nommé `original.txt` dans le répertoire courant, et que nous voulions créer un lien symbolique vers celui-ci appelé `softlink`. Nous pourrions utiliser :

```
$ ln -s original.txt softlink
```

Et apparemment, tout irait bien. Vérifions avec `ls` :

```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol  12 Jun  7 19:23 softlink -> original.txt
```

Voyez comment le lien est construit : le `softlink` pointe vers (`->`) `original.txt`. Cependant, voyons ce qui se passe si nous déplaçons le lien vers le répertoire parent et essayons d'afficher son contenu en utilisant la commande `less` :

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

Comme le chemin d'accès au fichier `original.txt` n'a pas été spécifié, le système suppose qu'il se trouve dans le même répertoire que le lien. Lorsque cela n'est plus vrai, le lien cesse de fonctionner.

La façon d'éviter cela est de toujours spécifier le chemin complet vers la cible lors de la création du lien :

```
$ ln -s /home/carol/Documents/original.txt softlink
```

Ainsi, quel que soit l'endroit où vous déplacez le lien, il fonctionnera toujours, car il indique l'emplacement absolu de la cible. Vérifiez avec `ls` :

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol  40 Jun  7 19:34 softlink -> /home/carol/Documents/original.txt
```



## Exercices Guidés

1. Imaginez qu'un programme ait besoin de créer un fichier temporaire à usage unique qui ne sera plus jamais nécessaire après la fermeture du programme. Quel serait le répertoire correct dans lequel créer ce fichier ?

2. Quel est le répertoire temporaire qui *doit* être effacé pendant le processus de démarrage ?

3. Quel est le paramètre de `chmod` en mode *symbolique* pour activer le sticky bit sur un répertoire ?

4. Imaginez qu'il y ait un fichier nommé `document.txt` dans le répertoire `/home/carol/Documents`. Quelle est la commande pour créer un lien symbolique vers ce fichier nommé `text.txt` dans le répertoire courant ?

5. Expliquez la différence entre un lien solide vers un fichier et une copie de ce fichier.

## Exercices d'Exploration

1. Imaginez que dans un répertoire, vous créez un fichier appelé `recipes.txt`. Dans ce répertoire, vous créez également un lien solide vers ce fichier, appelé `receitas.txt`, et un lien symbolique (ou *souple*) vers ce fichier appelé `rezepte.txt`.

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s recipes.txt rezepte.txt
```

Le contenu du répertoire doit apparaître ainsi :

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol 12 jun 24 10:12 rezepte.txt -> receitas.txt
```

N'oubliez pas que, en tant que lien solide, `receitas.txt` pointe vers le même inode que `recipes.txt`. Qu'arriverait-il au lien symbolique `rezepte.txt` si le nom `receitas.txt` était supprimé ? Pourquoi ?

2. Imaginez une clé USB branchée sur votre système, et montée sur `/media/votreutilisateur/FlashA`. Vous voulez créer dans votre répertoire personnel un lien appelé `schematics.pdf`, pointant sur le fichier `esquema.pdf` dans le répertoire racine de la clé USB. Alors, vous tapez la commande :

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

Que se passerait-il ? Pourquoi ?

3. Considérons le résultat suivant de `ls -lah` :

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol 77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
```

```
-rw-r--r-- 4 carol carol 77K jun 17 17:25 text.txt
```

- Combien de liens pointent vers le fichier `document.txt` ?
- S'agit-il de liens symboliques ou solides ?
- Quel paramètre devez-vous passer à `ls` pour voir quel inode occupe chaque fichier ?

4. Imaginez que vous ayez dans votre répertoire `~/Documents` un fichier nommé `clients.txt` contenant quelques noms de clients, et un répertoire nommé `somedir`. À l'intérieur de celui-ci, il y a un *autre* fichier *également* nommé `clients.txt` avec des noms différents. Pour reproduire cette structure, utilisez les commandes suivantes.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Vous créez ensuite un lien dans `somedir` nommé `partners.txt` qui pointe vers ce fichier, avec les commandes :

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

La structure du répertoire est donc la suivante :

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Maintenant, vous déplacez `partners.txt` de `somedir` vers `~/Documents`, et vous en listez le contenu.

```
$ cd ~/Documents/
```

```
$ mv somedir/partners.txt .  
$ less partners.txt
```

Le lien fonctionnera-t-il encore ? Si oui, quel fichier sera listé ? Pourquoi ?

5. Examinez les fichiers suivants :

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt  
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quelles sont les permissions d'accès de `partners.txt` ? Pourquoi ?

## Résumé

Dans cette leçon, vous avez appris :

- Où sont stockés les fichiers temporaires.
- Quelle est la permission spéciale qui leur est appliquée.
- Ce que sont les liens.
- La différence entre les liens *symboliques* et les liens *solides*.
- Comment créer des liens.
- Comment les déplacer, les renommer ou les supprimer.

Les commandes suivantes ont été abordées dans cette leçon :

- `ln`
- Le paramètre `-i` de `ls`

## Réponses aux Exercices Guidés

1. Imaginez qu'un programme ait besoin de créer un fichier temporaire à usage unique qui ne sera plus jamais nécessaire après la fermeture du programme. Quel serait le répertoire correct dans lequel créer ce fichier ?

Comme nous ne nous soucions pas du fichier une fois que le programme a fini de s'exécuter, le répertoire correct est `/tmp`.

2. Quel est le répertoire temporaire qui *doit* être effacé pendant le processus de démarrage ?

Ce répertoire est `/run` ou, sur certains systèmes, `/var/run`.

3. Quel est le paramètre de `chmod` en mode *symbolique* pour activer le sticky bit sur un répertoire ?

Le symbole du sticky bit en mode symbolique est `t`. Comme nous voulons activer (ajouter) cette permission au répertoire, le paramètre doit être `+t`.

4. Imaginez qu'il y ait un fichier nommé `document.txt` dans le répertoire `/home/carol/Documents`. Quelle est la commande pour créer un lien symbolique vers ce fichier nommé `text.txt` dans le répertoire courant ?

`ln -s` est la commande pour créer un lien symbolique. Puisque vous devez spécifier le chemin complet du fichier auquel vous vous connectez, la commande est :

```
$ ln -s /home/carol/Documents/document.txt text.txt
```

5. Expliquez la différence entre un lien solide vers un fichier et une copie de ce fichier.

Un lien solide n'est qu'un autre nom pour un fichier. Même s'il ressemble à une copie du fichier original, à toutes fins utiles, le lien et l'original sont les mêmes, car ils pointent vers les mêmes données sur le disque. Les modifications apportées au contenu du lien seront répercutées sur l'original, et vice-versa. Une copie est une entité complètement indépendante, occupant une place différente sur le disque. Les modifications apportées à la copie ne seront pas répercutées sur l'original, et vice-versa.

## Réponses aux Exercices d'Exploration

1. Imaginez que dans un répertoire, vous créez un fichier appelé `recipes.txt`. Dans ce répertoire, vous créez également un lien solide vers ce fichier, appelé `receitas.txt`, et un lien symbolique (ou *souple*) vers ce fichier appelé `rezepte.txt`.

```
$ touch recipes.txt
$ ln recipes.txt receitas.txt
$ ln -s recipes.txt rezepte.txt
```

Le contenu du répertoire devrait être ainsi :

```
$ ls -lhi
total 160K
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
5388837 lrwxrwxrwx 1 carol carol 12 jun 24 10:12 rezepte.txt -> recipes.txt
```

N'oubliez pas que, en tant que lien solide, `receitas.txt` pointe vers le même inode que `recipes.txt`. Qu'arriverait-il au lien symbolique `rezepte.txt` si le nom `receitas.txt` était supprimé ? Pourquoi ?

Le lien symbolique `rezepte.txt` cesserait de fonctionner. En effet, les liens symboliques pointent vers des noms, et non des inodes, et le nom `receitas.txt` n'existe plus, même si les données sont toujours sur le disque sous le nom `recipes.txt`.

2. Imaginez une clé USB branchée sur votre système, et montée sur `/media/votreutilisateur/FlashA`. Vous voulez créer dans votre répertoire personnel un lien appelé `schematics.pdf`, pointant sur le fichier `esquema.pdf` dans le répertoire racine de la clé USB. Alors, vous tapez la commande :

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

Que se passerait-il ? Pourquoi ?

La commande échouerait. Le message d'erreur serait "Invalid cross-device link", et il en indique clairement la raison : les liens solides ne peuvent pas pointer vers une cible dans une partition ou un périphérique différent. La seule façon de créer un lien de ce type est d'utiliser un lien *symbolique* ou *souple*, en ajoutant le paramètre `-s` à `ln`.

### 3. Considérons le résultat suivant de `ls -lah` :

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol 77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol 77K jun 17 17:25 text.txt
```

- Combien de liens pointent vers le fichier `document.txt` ?

Chaque fichier commence avec un nombre de liens égal à 1. Comme le nombre de liens pour le fichier est de 4, il y a trois liens qui pointent vers ce fichier.

- S'agit-il de liens symboliques ou solides ?

Il s'agit de liens solides, puisque les liens symboliques n'augmentent pas le nombre de liens d'un fichier.

- Quel paramètre devez-vous passer à `ls` pour voir quel inode occupe chaque fichier ?

Le paramètre est `-li`. L'inode apparaîtra dans la première colonne de la sortie de `ls`, comme ci-dessous :

```
$ ls -lahi
total 3,1M
5388773 drwxr-xr-x 2 rigues rigues 4,0K jun 17 17:27 .
5245554 drwxr-xr-x 5 rigues rigues 4,0K jun 17 17:29 ..
5388840 -rw-rw-r-- 1 rigues rigues 2,8M jun 17 15:45 compressed.zip
5388833 -rw-r--r-- 4 rigues rigues 77K jun 17 17:25 document.txt
5388837 -rw-rw-r-- 1 rigues rigues 216K jun 17 17:25 image.png
5388833 -rw-r--r-- 4 rigues rigues 77K jun 17 17:25 text.txt
```

4. Imaginez que vous ayez dans votre répertoire `~/Documents` un fichier nommé `clients.txt` contenant quelques noms de clients, et un répertoire nommé `somedir`. À l'intérieur de celui-ci, il y a un *autre* fichier *également* nommé `clients.txt` avec des noms différents. Pour reproduire cette structure, utilisez les commandes suivantes.

```
$ cd ~/Documents
```



```
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Vous créez ensuite un lien dans `somedir` nommé `partners.txt` qui pointe vers ce fichier, avec les commandes :

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

La structure du répertoire est donc la suivante :

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Maintenant, vous déplacez `partners.txt` de `somedir` vers `~/Documents`, et vous en listez le contenu.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

Le lien fonctionnera-t-il encore ? Si oui, quel fichier sera listé ? Pourquoi ?

C'est un cas "délicat", mais le lien fonctionnera, et le fichier listé sera celui de `~/Documents`, contenant les noms `John, Michael, Bob`.

N'oubliez pas que, puisque vous n'avez pas spécifié le chemin complet vers le fichier `clients.txt` cible lors de la création du lien souple `partners.txt`, l'emplacement cible sera interprété comme étant relatif à l'emplacement du lien, qui dans ce cas est le répertoire courant.

Lorsque le lien a été déplacé de `~/Documents/somedir` vers `~/Documents`, il devrait cesser de fonctionner, puisque la cible n'est plus dans le même répertoire que le lien. Cependant, il se trouve qu'il y a un fichier nommé `clients.txt` dans `~/Documents`, donc le lien pointera vers ce fichier, au lieu de la cible originale dans `~/somedir`.

Pour éviter cela, il faut toujours spécifier le chemin complet vers la cible lors de la création d'un lien symbolique.

5. Examinez les fichiers suivants :

```
-rw-r--r-- 1 rigues rigues 19 Jun 24 11:12 clients.txt  
lrwxrwxrwx 1 rigues rigues 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quelles sont les permissions d'accès de `partners.txt` ? Pourquoi ?

Les permissions d'accès pour `partners.txt` sont `rw-r--r--`, car les liens héritent toujours des mêmes permissions d'accès que la cible.

## Impression

© 2024 par Linux Professional Institute: Supports de cours, “Linux Essentials (Versión 1.6)”.

PDF généré: 2024-04-29

Cette oeuvre est placée sous licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International (CC-BY-NC-ND 4.0). Pour consulter une copie de cette licence, visitez

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Bien que le Linux Professional Institute ait fait tout son possible pour s'assurer que les informations et les instructions contenues dans cette publication soient exactes, le Linux Professional Institute décline toute responsabilité en cas d'erreurs ou d'omissions, y compris, mais sans s'y limiter, la responsabilité des dommages résultant de l'utilisation ou de la confiance accordée à cette publication. L'utilisation des informations et des instructions contenues dans cet ouvrage se fait à vos risques et périls. Si les exemples de code ou toute autre technologie décrite ou contenue dans cet ouvrage sont soumis à des licences open source ou aux droits de propriété intellectuelle de tiers, il vous incombe de veiller à ce que leur utilisation soit conforme à ces licences et/ou ces droits.

Les supports de cours du LPI sont une initiative du Linux Professional Institute (<https://lpi.org>). Les supports de cours et leurs traductions sont disponibles à l'adresse <https://learning.lpi.org>.

Pour toute question ou commentaire sur cette édition ou sur l'ensemble du projet, contactez-nous à l'adresse suivante : [learning@lpi.org](mailto:learning@lpi.org).