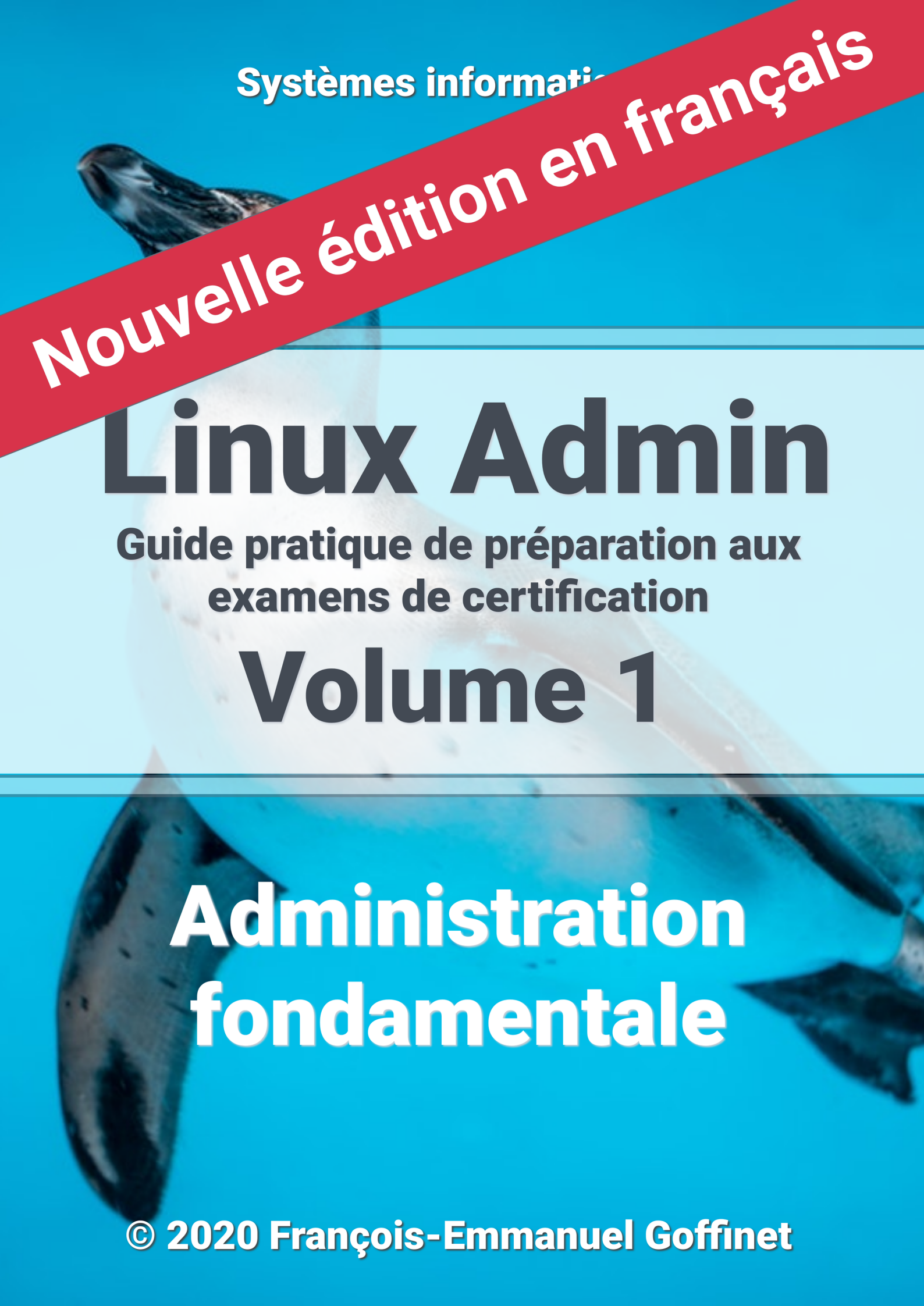

Systèmes informati
Nouvelle édition en français
Linux Admin
Guide pratique de préparation aux examens de certification
Volume 1
Administration fondamentale
© 2020 François-Emmanuel Goffinet

Linux Administration Volume 1

Linux Administration Fondamentale. Guide pratique de préparation aux examens de certification LPIC 1, Linux Essentials, RHCSA et LFCS. Administration fondamentale. Introduction à Linux. Le Shell. Traitement du texte. Arborescence de fichiers. Sécurité locale.

François-Emmanuel Goffinet

Ce livre est en vente à http://leanpub.com/linux-administration-volume-1

Version publiée le 2021-09-06

Leanpub

Ce livre est publié par Leanpub. Leanpub permet aux auteurs et aux éditeurs de bénéficier du Lean Publishing. Lean Publishing consiste à publier à l'aide d'outils très simples de nombreuses itérations d'un livre électronique en cours de rédaction, d'obtenir des retours et commentaires des lecteurs afin d'améliorer le livre.

Aussi par François-Emmanuel Goffinet

Cisco CCNA 200-301 Volume 1

Cisco CCNA 200-301 Volume 2

Cisco CCNA 200-301 Volume 3

Cisco CCNA 200-301 Volume 4

Linux Administration Volume 2

Linux Administration Volume 3

Linux Administration Volume 4

Protocole SIP

Table des matières

Avertissement

Droits

Ce document s'inspire de près ou de loin de toute une série d'autres qui sont soumis la plupart du temps aux mêmes droits. Les sources citées ou reprises sont présentes sous format d'URI dans le code source. J'espère que les auteurs concernés se satisferont de cette exposition. Les marques citées ont été déposées par leurs propriétaires.

Dédicace

À S., amoureusement et à R., tendrement.

Remerciements

Merci aux milliers de visiteurs quotidiens du site linux.goffinet.org.

Merci aux centres de formation et aux écoles qui m'accordent leur confiance et qui me permettent de rencontrer mon public en personne.

Merci à Linux Torvalds qui mérite une reconnaissance universelle pour le don de son oeuvre à l'humanité.

Avant-Propos

François-Emmanuel Goffinet est formateur IT et enseignant depuis 2002 en Belgique et en France. Outre Cisco CCNA, il couvre de nombreux domaines des infrastructures informatiques, du réseau à la virtualisation des systèmes, du nuage à la programmation d'infrastructures hétérogènes en ce y compris DevOps, Docker, K8s, chez AWS, GCP ou Azure, etc. avec une forte préférence et un profond respect pour l'Open Source, notamment pour Linux.

On trouvera ici un des résultats d'un projet d'autopublication en mode *agile* plus large lié au site web linux.goffinet.org.

Ce document fait partie d'un guide de formation en français sur les pratiques sécurisées d'administration du système d'exploitation (OS) GNU/Linux. Le propos invite progressivement à déployer les technologies de virtualisation, à procéder à des tâches d'automation / automatisation via des scripts, à déployer les services traditionnels tels que des services Web ou d'infrastructure, voire plus spécifiques en ToIP / VoIP / UC ou même IaaS.

Le document comprend de nombreux scripts et exemples. Aussi, il traite les sujets sur les distributions basées RHEL (Centos et dérivés) et Debian Stable (Ubuntu et autres dérivés).

Le document vise à atteindre un double objectif :

- Maintenir de manière durable un cours transversal, réutilisable librement, ouvert et actualisé sur les systèmes fonctionnant sous GNU/Linux.
- Aligner les contenus et les pratiques décrites dans les programmes des certifications LPI (Linux Profesional Institute), RH (Red Hat) et LFS (Linux Foundation Software), etc.

Orientation pédagogique

Ce document oriente le contenu sur :

- La virtualisation, les technologies en nuage (*cloud*)
- L'automatisation par la rédaction de code (scripts)
- Les pratiques de sécurité

Public cible du document

Ce document s'adresse à tous les professionnels de l'informatique bien sûr mais aussi des services et de l'industrie pour lesquels l'ère numérique a modifié les pratiques de travail.

Du bon usage du support

Ce support évolue constamment selon l'épreuve du temps et des retours d'expérience. Il est toujours préférable de se référer à la dernière version en ligne sur https ://linux.goffinet.org.

Il se lit ou s'expose en face d'une **console Linux**, dans une machine virtuelle par exemple. Les interfaces graphiques des logiciels seront laissées à l'appréciation des utilisateurs.

Pour obtenir de meilleurs résultats d'apprentissage, notamment en classe de formation, il est conseillé d'utiliser une **installation native**, avec une ligne de commande ou un *shell* à disposition.

Enfin, ce document n'étant qu'un support de cours, il sera nécessaire de visiter les références et les liens fournis ainsi que les sites officiels et leurs pages de documentation qui restent dans la plupart des cas librement disponibles.

Distributions de référence

On conseillera quelques distributions Linux de référence avant d'entamer des distributions spécialisées ou spécifiques.

1. Centos / (RHEL) / Fedora
2. Ubuntu 20.04 LTS Focal / Debian Stable

Matériel nécessaire

Un ordinateur individuel récent connecté au réseau local (et à l'Internet) est nécessaire. Dans une classe de formation, la meilleure expérience est d'installer une distribution Linux native et d'utiliser des outils de virtualisation tels que *libvirt* et *qemu/KVM* pour réaliser des exercices avancés.

Introduction

Ce premier volume Linux Administration fondamentale s'aligne sur les objectifs Linux Essentials, LPIC 1, RHCSA et LFCS.

Il peut occuper une activité intellectuelle de 16 à 35 heures, voir plus.

L'objectif opérationnel est de prendre la main sur une console Linux dans une première approche fondamentale.

Une première partie a été conçue comme **journée d'introduction** à cours d'Administration Linux de 5 jours. Quand le temps fait défaut, si les conditions le permettent, ce sujet peut être appris par soi-même. Les quatre premiers chapitres *Evolution de Linux*, *Distributions Linux et cycles de maintenance*, *Licences Open Source* et *Applications Open Source* sont des présentations de l'écosystème Linux. Les deux derniers chapitres *Utiliser Linux en console graphique* et *Environnements de bureau* sont des exercices pratiques d'installation, d'usage et de paramétrage d'une distribution Linux graphique (Desktop).

La seconde partie sur le Shell est une étape de prise en main de la console d'un système Linux. On y trouve un chapitre d'initiation à la ligne de commande, un chapitre sur le filtrage de fichiers, un exercice de création d'un script Shell, un chapitre sur la configuration des langues, locales et claviers, un chapitre qui nous apprend comment obtenir de l'aide sur un système Linux et enfin un dernier chapitre indique comment prendre connaissance de la version de sa distribution Linux.

La troisième partie intitulée "Traitement du texte" s'intéresse aux différents outils Unix de traitement du texte. Un premier chapitre s'intéresse aux outils natifs avec les tubes et les redirections avec les outils comme cat, tac, head, tail, tee, wc, tr, od, join, cut, sort, paste, diff. Un second chapitre s'intéresse aux expressions rationnelles (regexp) et aux outils grep, sed et awk. Enfin un dernier chapitre est consacré à l'éditeur vi.

Une quatrième partie intitulée "Arborescence de fichiers" permet dans un premier temps d'appréhender la structure d'un système de fichier Linux en s'y déplaçant. Un second chapitre traite des opérations sur les fichiers (inodes, listing, création de fichiers, de répertoires, déplacements, copies, effacement, liens). Un troisième chapitre se consacre entièrement à la pratique de recherche de fichier sur un système Linux. Enfin un dernier chapitre traitre des outils d'archivage et de compression de fichiers comme gzip, bzip2, tar ou zip.

Enfin, l'ouvrage se termine par une partie intitulée "Sécurité locale" s'intéresse aux utilisateurs et groupes Linux ainsi que tous les éléments qui concernent leur gestion : mots de passe, connexions su et sudo, opérations sur les utilisateurs et groupes (création, ajout, désactivation, effacement), les permissions (rwx, ugo, SUID, SGID, Sticjy bit), les ACLs sur les fichiers et enfin l'usage de PAM.

Première partie Introduction à Linux

Objectifs de certification

Objectifs Linux Essentials

Ce chapitre est une introduction à un cours Linux Essentials dont les objectifs sont vérifiés dans la certification LPI 117-010.

Les objectifs abordés ici sont :

- *Sujet 1 : Communauté Linux et carrière dans le logiciel libre*
  - 1.1 Évolution de Linux et systèmes d'exploitation populaires (valeur : 2)
  - 1.2 Applications libres majeures (valeur : 2)
  - 1.3 Logiciel à code source ouvert et licences (valeur : 1)
  - 1.4 Compétences informatiques et travail sous Linux (valeur : 2)
- *Sujet 4 : Le système d'exploitation Linux*
  - 4.1 Choix d'un système d'exploitation (valeur : 1)

Objectifs LPIC 1

- *Sujet 102 : Installation de Linux et gestion de paquetages*
  - 102.1 Conception du schéma de partitionnement
  - 102.2 Installation d'un gestionnaire d'amorçage
- *Sujet 106 : Interfaces et bureaux utilisateur*

Introduction

Cette partie a été conçue comme **journée d'introduction** à cours d'Administration Linux de 5 jours. Quand le temps fait défaut, si les conditions le permettent, ce sujet peut être appris par soi-même.

Les quatre premiers chapitres *Evolution de Linux*, *Distributions Linux et cycles de maintenance*, *Licences Open Source* et *Applications Open Source* sont des présentations de l'écosystème Linux. Les deux derniers chapitres *Utiliser Linux en console graphique* et *Environnements de bureau* sont des exercices pratiques d'installation, d'usage et de paramétrage d'une distribution Linux graphique (Desktop).

En classe de formation, il est conseillé de procéder à une installation native (directement sur le matériel).

1. Évolution de Linux

Objectifs de certification

Objectifs Linux Essentials 1.1

- **Connaissance du développement de Linux et des distributions majeures**.
- Domaines de connaissance les plus importants :
  - Philosophie des Logiciels libres.
  - Distributions.
  - Systèmes embarqués.
- Liste partielle de termes, fichiers et utilitaires utilisés pour cet objectif :
  - Android.
  - Debian.
  - CentOS.

1. Système d'exploitation (OS)

- Une des tâches du système d'exploitation est d'offrir aux utilisateurs une interface simple et conviviale avec le matériel.
- Un système d'exploitation (souvent appelé OS pour Operating System) est un ensemble de programmes qui dirige l'utilisation des capacités d'un ordinateur (matériel) par des logiciels applicatifs.

1.1. Fonction d'un OS

Il s'occupe au minimum de :

- La gestion des processus (programmes)
- La gestion de la mémoire
- Le système de fichiers
- La gestion des entrées/sorties

1.2. Caractéristiques d'un OS moderne

Linux est un système d'exploitation :

- **Multi-tâches** : Un système d'exploitation est multitâche (en anglais : multitasking) s'il permet d'exécuter, de façon apparemment simultanée, plusieurs programmes informatiques.
- **Multi-utilisateurs** : Un système d'exploitation multi-utilisateur est conçu pour permettre à plusieurs utilisateurs d'utiliser l'ordinateur simultanément, tout en limitant les droits d'accès de chacun afin de garantir l'intégrité de leurs données.
- **Multi-processeurs**

2. Architectures matérielles

Linux est supporté sur tout type d'architecture :

- Serveurs d'entreprise
- Serveurs de Data Center
- Ordinateurs de bureau
- Ordinateurs portables
- Ordinateurs légers
- Mainframes
- Embarqués Industrie
- Embarqués automobile, domotique, domestique, …
- Appareils mobiles, appareils légers
- CPE,
- Périphériques d'infrastructure réseau/stockage/multimédia

2.1. Processeurs

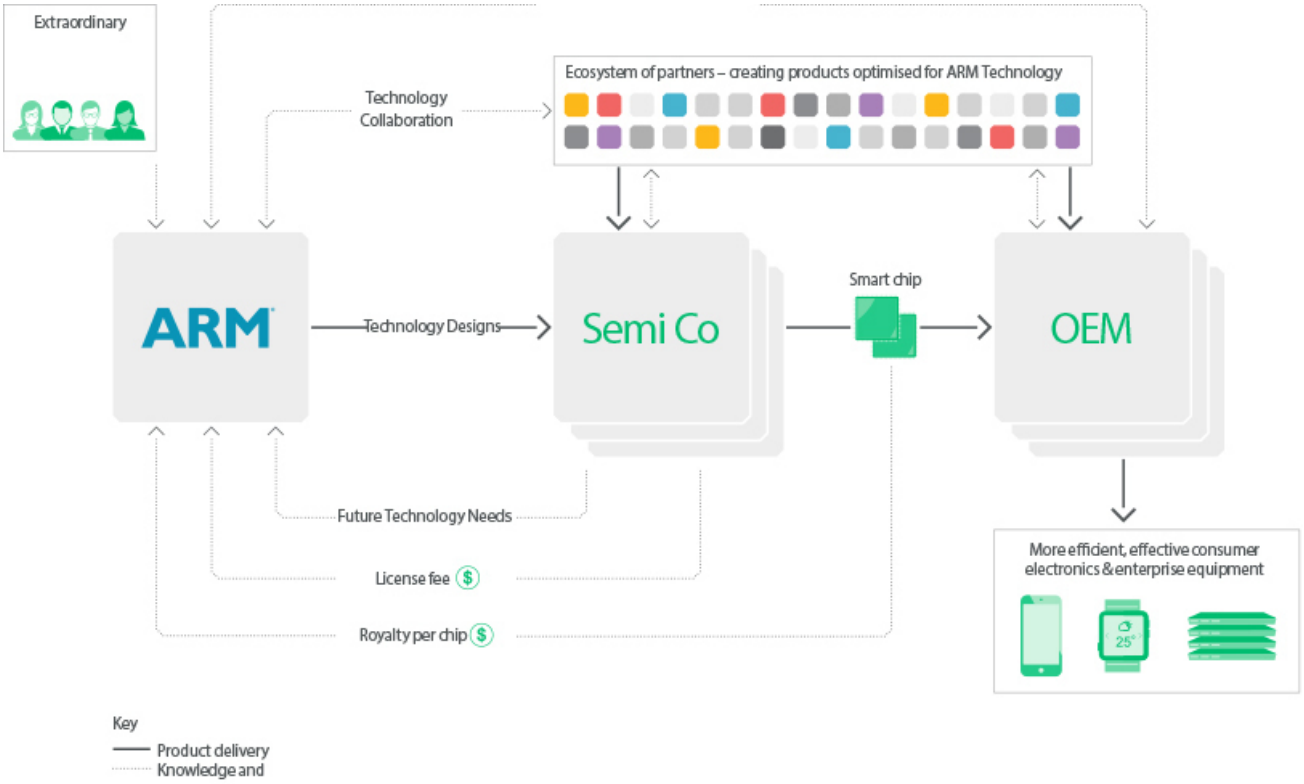
ARM	Intel/AMD
Architecture RISC	Architecture CISC
A performance égale, réduction des coûts de production et meilleure efficacité thermique (ARM Cortex-A15 28nm 1.62mm²)	Complexité matérielle plus coûteuse (AMD Jaguar 28nm 3.1mm²) en conception et en énergie
Stratégie commerciale : licence	Intel/AMD
Unix	Unix / Windows
Bootloader	Bios

2.2. ARM Business Model

Les architectures ARM sont des architectures matérielles RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8)1 développées par ARM Ltd depuis 1990 et introduites à partir de 1983 par Acorn Computers.

Dotés d'une architecture relativement plus simple que d'autres familles de processeurs, et bénéficiant d'une faible consommation, les processeurs ARM sont devenus dominants dans le domaine de l'informatique embarquée, en particulier la téléphonie mobile et les tablettes.

Ces processeurs sont fabriqués sous licence par un grand nombre de constructeurs.



Source : https ://ir.arm.com/phoenix.zhtml ?c=197211&p=irol-model et https ://fr.wikipedia.org/wiki/Architecture_-ARM

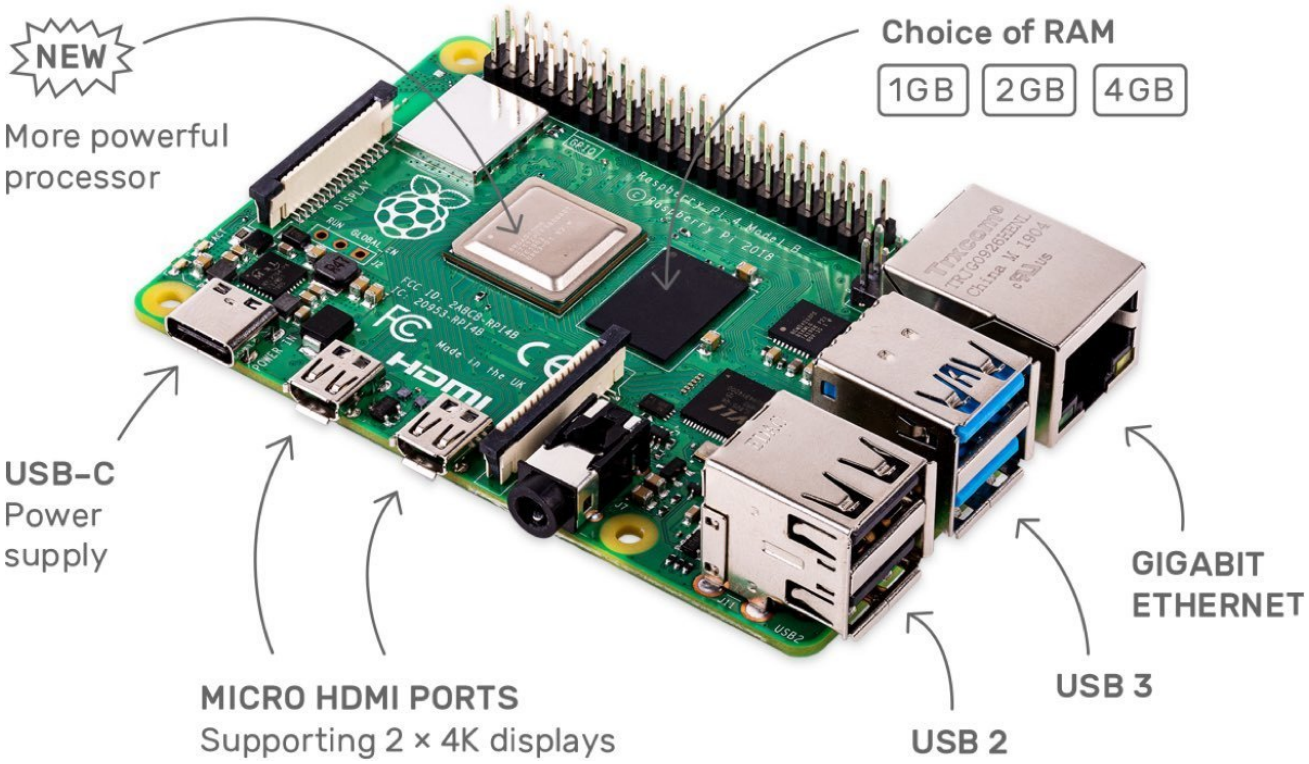
2.3. Matériel embarqué

On trouvera ici une liste des ordinateurs embarqués dans https ://en.wikipedia.org/wiki/Comparison_of_single-board_computers dans laquelle on retrouve des plateformes ARM, Intel et AMD, mais aussi des architectures MIPS.

À titre d'exemple, Openwrt est une distribution Linux pour routeurs domestiques dont on trouve ici la liste.

Embarqué domestique : Raspberry Pi

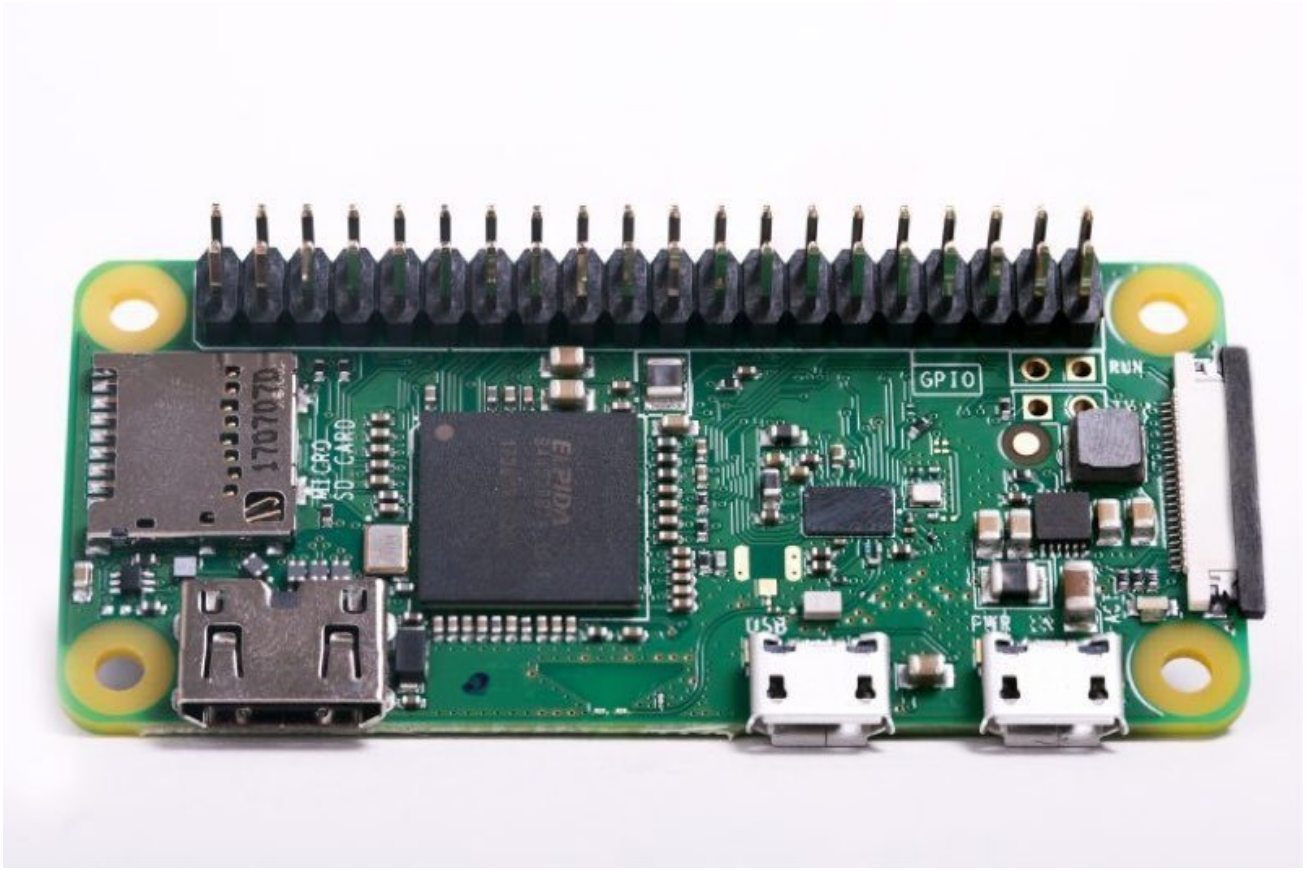
Les plateformes Raspberry Pi sont très populaires. Leur système d'explotation favori est une distribution Linux dont la distribution Raspbian (dérivé Debian) qui est la distribution officiellle et qui dispose d'un communauté très large.



Raspberry Pi 4 B

Par exemple, Raspberry Pi 4, Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, 1Go/2Go/4Go RAM, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE intégrés, Gigabit Ethernet, 2 USB 3.0 ports ; 2 USB 2.0 ports, 2 × micro-HDMI ports, 5V DC via USB-C connector (minimum 3A*) ou via GPIO header. Source : https ://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

Autre exemple, comme le Raspberry Pi Zero WH, 1 CPU Broadcom BCM2835 candencé à 1GHz ARM11, 512MB RAM, Micro-USB sockets for data and power, 2.4GHz 802.11 b/g/n Wi-Fi, Bluetooth 4.1 LE, 40 pin GPIO header soudé. Il mesure 30X56X13 mm, il pèse 12 g. et il consomme 180 mA seulement !



Raspberry Pi Zero WH

Pour comparer les différents modèles Raspberry Pi : http ://socialcompare.com/en/comparison/raspberrypi-models-comparison.

Embarqué industriel : PC Engines, Mirabox

Globalscale Mirabox

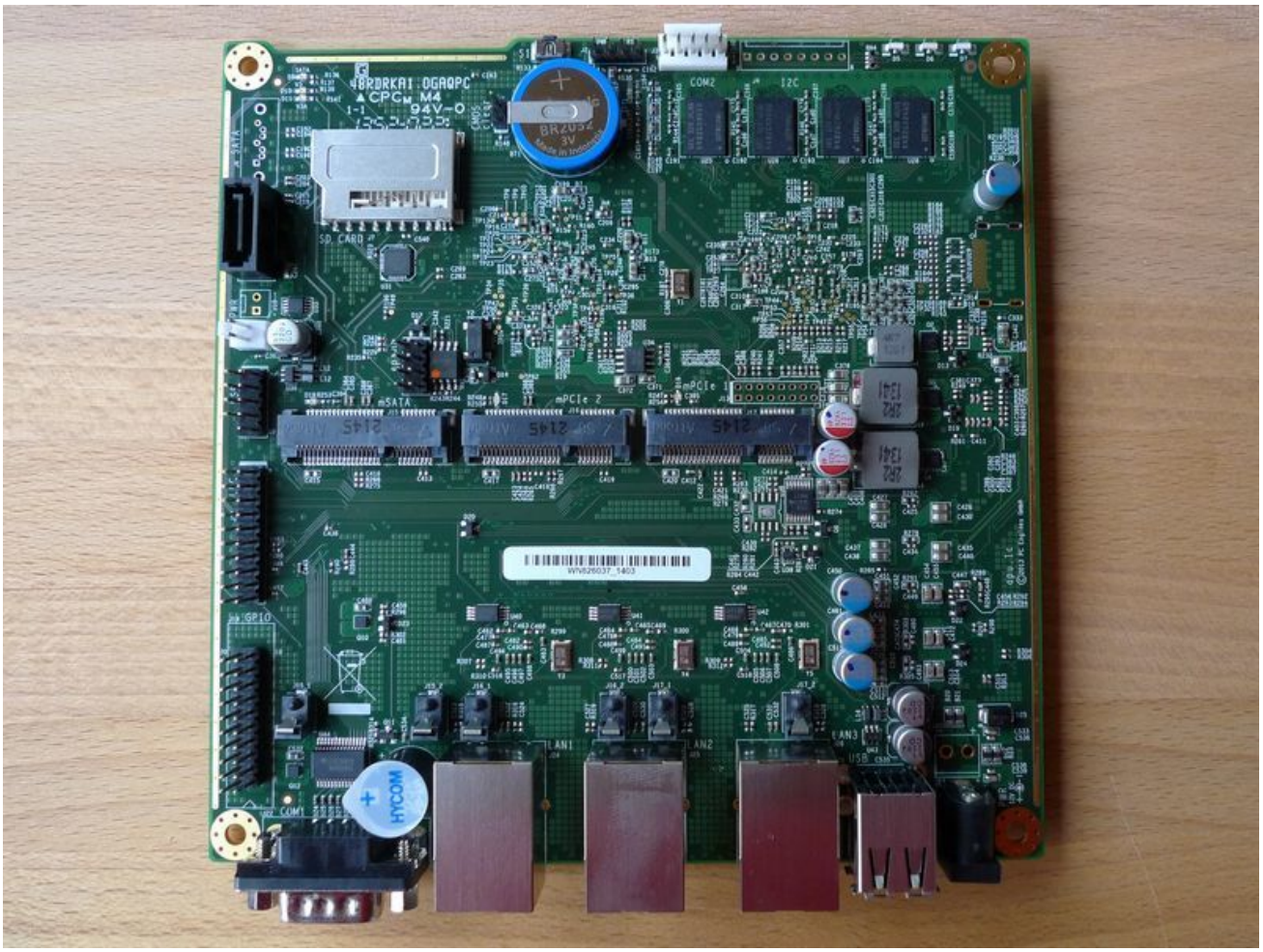
Gobalscale Mirabox, Source : https ://wiki.ipfire.org/en/hardware/arm/globalscale/mirabox

- 1.2Ghz Marvell Armada CPU ARMADA 370 ARM v7 compliant

- 802.11b/g/n Wifi with Marvell 88W8787 and Bluetooth 3.0
- 1GB DDR3
- 1 GB NAND Flash
- 2 each 10/100/1000 Ethernet Ports
- 2 each USB 3.0 host
- 1 microsd card slot/reader, 1 additional Mini PCIe slot for expansion (internal) For additional 2x2, 3x3, 4x4 WiFi Radios, or 3G modules
- 3 LED controlled by GPIO, reset button
- external power supply
- Port for JTAG and Debugging options

PC Engines développe et vend de petits ordinateurs monocartes pour la mise en réseau à une clientèle mondiale. La société a été fondée en 1995 à Sunnyvale, Californie. Nous avons commencé par concevoir des PC embarqués sur mesure pour divers clients OEM. PC Engines a déménagé en Suisse en 2002. Depuis lors, nous nous sommes concentrés sur l'offre de nos produits à marque propre.

Le matériel embarqué APU1D4 de PC Engines par exemple est une plateforme AMD X64 (1 GHz dual Bobcat core) avec 4Go RAM embarqués, trois interfaces Gigabit Ethernet, un port pour une carte SD, 1 port USB 2.0, un ports m-sata, deux ports emplacements miniPCI express et un emplacement pour héberger une carte SIM. Il est dimensionné en 152.4 x 152.4 mm.



APU1D4 PC Engines

Sources : https ://pcengines.ch/apu1d4.htm

Embarqué Hacking : Hak5

Hak5 est une entreprise américaine spécialisée dans la conception de matériel de Hacking pour des attaques de proximité :

Intrusion Wi-Fi

- Wifipineapple Nano
- Wifipineapple Tetra

Implants réseau filaire / sans-fil

- Lan Turtle
- Pacquet Squirrel
- Send Owl
- Plunder Bug (Tap Ethernet)

Attaques de proximité USB/Ethernet * Shark Jack * Rubber Ducky (contrôlleur) * Bash Bunny

Le matériel est optimisé pour la légèreté et la faible consommation d'énergie. Il est dédié à des tâches spécialisées d'audit de sécurité. Il dispose de commutateurs et de LED que l'on peut programmer nativement.

Toutes les plateformes de type "ordinateur" (sauf "Plunder Bug" et "Bash Bunny") fonctionnent sur une base de la distribution OpenWRT. Les charges d'audit sont conçues en Bash ou en Python. Les binaires de "pentesting" comme ncat, tcpdump, etc. et d'autres plus rares sont de fabrication "maison".

3. Qu'est-ce que Linux ?

- Linux est d'abord le nom d'un noyau (le contrôleur central)
- Avec quelques outils supplémentaires, on obtient un système d'exploitation (OS) :
  - Un environnement Shell (une ligne de commande)
  - La gestion du système (ajouter des utilisateurs,...)
  - Des applications (mail, web, développement,...)
- Le tout est mis dans une distribution Linux :
  - dépôts de paquetages, maintenance des logiciels, scripts de lancement,...
  - interfaces graphiques, communautés, ...

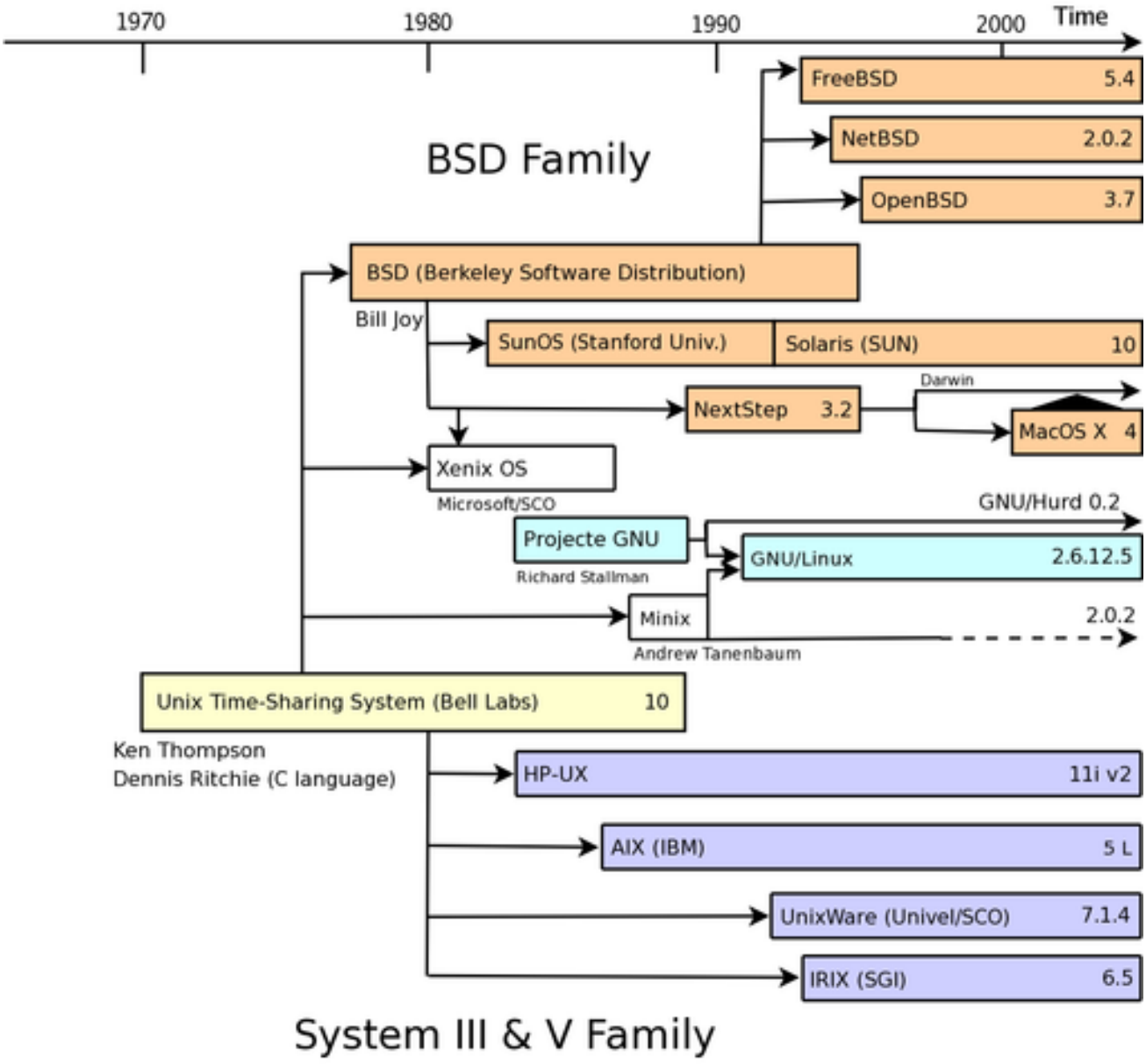
3.1. Origine de Linux

- Créé en 1991 par Linus Torvalds pour des processeurs 80386, il y a plus de 20 ans.
- Reproduit le comportement d'un noyau UNIX (1969).
- Repris par une communauté de développement.
- Le projet GNU ajoute une série d'outils autour du noyau.

3.2. Qu'est-ce que Unix ?

- Unix a été créé par Bell Labs en 1969.
- Populaire dans les milieux académiques et sur les Mainframes (1980).
- Donne le nom à une famille de systèmes d'exploitation (notamment FreeBSD, NetBSD et OpenBSD), Dalvik/Linux (Android), GNU/Linux, iOS et OS X.
- Le nom « UNIX » est une marque déposée de l'Open Group, qui autorise son utilisation pour tous les systèmes certifiés conformes à la *Single UNIX Specification*.

3.3. Famille Unix



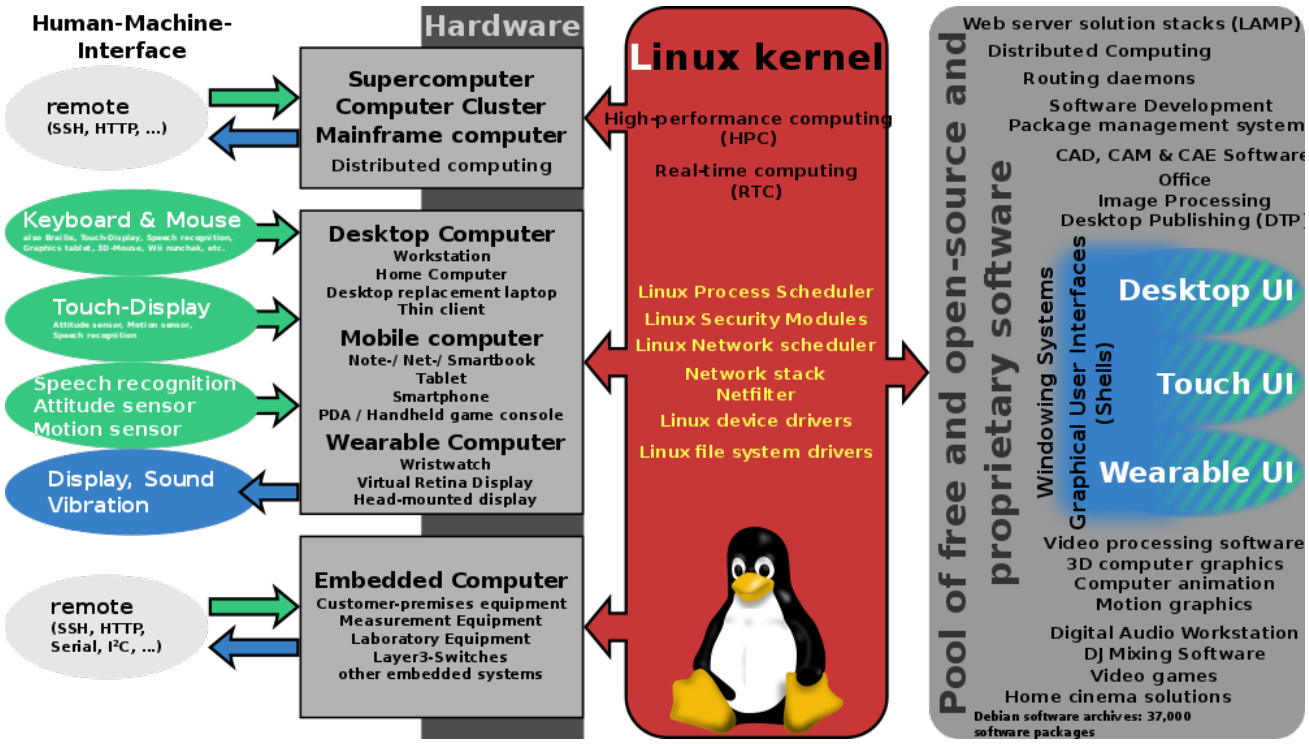
Famille Unix

Source de l'image Famille Unix

3.4. Que fait Linux ?

- Le noyau gère les processus applicatifs
- Attribue et récupère la mémoire
- Gère les accès aux disques et au processeur (CPU)
- Met une couche d'abstraction sur le matériel pour des applications "hardware-agnostic"
- Fournit la sécurité et l'isolation des utilisateurs
- Est capable de passer à la gestion de processus multiples (preemptive multitasking, SMP)

3.5. Ubiquité du noyau Linux



Ubiqiuité du noyau Linux.

Source l'image Ubiqiuité du noyau Linux.

4. GNU

GNU est un projet de système d'exploitation libre lancé en 1983 par Richard Stallman, puis maintenu par le projet GNU. Son nom est un acronyme récursif qui signifie en anglais « GNU's Not UNIX » (littéralement, « GNU n'est pas UNIX »). Il reprend les concepts et le fonctionnement d'UNIX. Le système GNU permet l'utilisation de tous les logiciels libres, pas seulement ceux réalisés dans le cadre du projet GNU. Son symbole est un gnou, un animal vivant en Afrique.

Il existe à ce jour deux distributions du système d'exploitation GNU :

- Arch Hurd ;
- Debian GNU/Hurd.

GNU/Linux (souvent appelé "Linux") est une variante du système d'exploitation GNU fonctionnant avec le noyau Linux. Le projet GNU avait originellement prévu le développement du noyau Hurd pour compléter le système, mais au début des années 1990, Hurd ne fonctionnait pas encore et son développement rencontrait encore des difficultés. L'arrivée du noyau Linux permit l'utilisation du système GNU sur les ordinateurs animés par des microprocesseurs de la famille Intel x86, en favorisant sa large diffusion par la complémentarité des projets.

Source : https ://fr.wikipedia.org/wiki/GNU

5. Evolution des OS

- Support de la virtualisation

- Support accru pour les architectures autres qu'Intel
- Support de la reconnaissance automatique du matériel
- Un support et un développement communautaire

6. Open Source

- Les êtres humains conçoivent des applications, des systèmes et des idées en langue intelligible pour les machines : du code à exécuter.
- Le terme "Open Source" au fait suivant : vous avez **accès** au code et que vous pouvez le **modifier**.

7. Références

- https ://fr.wikibooks.org/wiki/Le_syst%C3%A8me_d%27exploitation_GNU-Linux
- https ://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation
- https ://fr.wikipedia.org/wiki/Projet_GNU

Deuxième partie Le Shell

Objectifs de certification

Linux Essentials

- *Sujet 2 : Trouver son chemin sur un système Linux*
  - 2.1 Bases sur la ligne de commande (valeur : 3)
  - 2.2 Utilisation de la ligne de commande pour obtenir de l'aide (valeur : 2)

RHCSA EX200

- **1. Comprendre et utiliser les outils essentiels**
  - 1.1. Accéder à une invite shell et écrire des commandes avec la syntaxe appropriée
  - 1.3. Utiliser des expressions grep et régulières pour analyser du texte
  - 1.11. Localiser, lire et utiliser la documentation système, notamment les manuels, informations et fichiers dans `/usr/share/doc`

LPIC 1

- *Sujet 103 : Commandes GNU et Unix*
  - 103.1 Travail en ligne de commande
- *Sujet 107 : Tâches d'administration*
  - 107.3 Paramètres régionaux et langues

Introduction

Cette partie sur le Shell est une étape de prise en main de la console d'un système Linux. On y trouve un chapitre d'initiation à la ligne de commande, un chapitre sur le filtrage de fichiers, un exercice de création d'un script Shell, un chapitre sur la configuration des langues, locales et claviers, un chapitre qui nous apprend comment obtenir de l'aide sur un système Linux et enfin un dernier chapitre indique comment prendre connaissance de la version de sa distribution Linux.

2. La ligne de commande

Voici une première approche de la ligne de commande du Shell Linux. On apprendra ici à manipuler les commandes du système et des alias des commandes.

1. La ligne de commande

- La ligne de commande est un moyen simple d'interagir avec un ordinateur.
- Le shell interprète les commandes tapées au clavier.
- Le *prompt*, ou l'invite de commande, qui se termine par un `$` pour un utilisateur standard ou un `#` pour l'administrateur du système (désigné *root*), indique que le shell attend les commandes de l'utilisateur.
- Le shell est également un langage de programmation que l'on peut utiliser pour lancer des tâches automatiquement.
- Les programmes shell sont appelés par des scripts.

1.1. Définitions

- Le terminal = l'environnement d'entrée/sortie
- La console = terminal physique

Shell =

- "Interpréteur" de commande : lancer des commandes,
- Environnement : confort de l'utilisateur, sécurité
- Langage de programmation : fonctionnalités
- Traitement du texte
- Interface avec le noyau
- …

En informatique, le **shell** est le moyen le plus simple d'interagir avec le système d'exploitation. Sur Microsoft Windows, selon les versions, le shell est appelé par les commandes `CMD.EXE` ou `COMMAND.COM`, sans oublier `powershell.exe`. Sur Linux ou Unix, il existe une grande variété de shells comme `sh`, `bash`, `ksh`, etc. Un shell accepte des commandes frappées à partir d'une invite de commandes (l'entrée standard) et les exécute, généralement en temps réel ; l'affichage des résultats est rendu en général sur un écran (la sortie standard).

1.2. Types de shells

On obtient la liste des shells présents sur le système en affichant le fichier `/etc/shells` :

```
cat /etc/shells
```

- **`sh`** : Bourn Shell, historique, standard, "portable"
- **`csh`**/**`tcsh`** : C Shell
- **`ksh`** : Korn Shell
- **`bash`** : Bourn Again Shell Linux, le plus utilisé

1.3. Normes

- POSIX
- Single Unix Specification (SUS).

1.4. Bourne Again SHell

Le projet GNU offre des outils pour l'administration de système de type UNIX qui sont libres et qui respectent les standards UNIX.

Bash est un Shell compatible avec sh qui incorpore des spécificités utiles du Korn Shell (ksh) et du C Shell (csh). Il est censé se conformer à la norme IEEE POSIX P1003.2/ISO 9945.2 Standards des Shell et Outils. Il offre des améliorations fonctionnelles par rapport à sh pour la programmation et l'utilisation interactive.

1.5. Le shell interactif

Quand on obtient un terminal avec une ligne de commande, on se situe dans un environnement encadré par un programme "shell" qui a créé un processus sur le système. Il permet notamment d'exécuter des commandes.

1.6. Commande echo

```
ECHO(1)                 Manuel de l'utilisateur Linux                ECHO(1)

NOM
       echo - Afficher une ligne de texte

SYNOPSIS
       echo [-neE] [message ...]
       echo {--help,--version}

DESCRIPTION
       Cette page de manuel documente la version GNU de echo.

       La  plupart  des  shells ont une commande intégrée ayant le même nom et
       les mêmes fonctionnalités.

       echo écrit chaque message sur la sortie standard, avec une espace entre
       chacun d'eux, et un saut de ligne après le dernier.
```

La commande `echo` permet d'afficher du texte à l'écran :

```
[francois@c7li ~]$ echo "affiche ce texte"
affiche ce texte
[francois@c7li ~]$
```

1.7. Prompt, Invite de commande

```
[francois@c7li ~]$
```

On remarque le "prompt", "l'invite de commande" composée de :

- `francois` l'utilisateur connecté
- `@` séparateur
- `c7li` nom de l'ordinateur
- ~ "titld" qui indique le dossier utilisateur comme dossier courant
- `$` qui indique le type de connexion

Cette configuration de l'environnement est chargée sous forme de script au moment de la connexion de l'utilisateur.

1.8. Commande `ls`

```
LS(1)                    Manuel de l'utilisateur Linux                    LS(1)

NOM
       ls, dir, vdir - Afficher le contenu d'un répertoire

SYNOPSIS
       ls [options] [fichier...]
       dir [fichier...]
       vdir [fichier...]

       Options POSIX : [-CFRacdilqrtu1] [--]

       Options GNU (forme courte) : [-1abcdfgiklmnopqrstuvwxABCDFGHLNQRSUX]
       [-w cols] [-T cols] [-I motif] [--full-time] [--show-control-chars]
       [--block-size=taille]      [--format={long,verbose,commas,across,verti□
       cal,single-column}]                [--sort={none,time,size,extension}]
       [--time={atime,access,use,ctime,status}] [--color[={none,auto,always}]]
       [--help] [--version] [--]

DESCRIPTION
       La commande ls affiche tout d'abord l'ensemble de ses arguments
       fichiers autres que des répertoires. Puis ls affiche l'ensemble des
       fichiers contenus dans chaque répertoire indiqué. Si aucun argument
       autre qu'une option n'est fourni, l'argument « . » (répertoire en
       cours) est pris par défaut.
```

2. Entrer des commandes dans l'invite

Pour entrer des commandes dans le shell, il faut :

- une **commande** valide (dans le PATH ou précisée par un chemin)

suivie *éventuellement* d'une ou plusieurs **options** notées

- par un "dash", le tiret, "-" en **notation abrégée** ou un double "dash", double tiret, "--" en **notation** extensive,
- des **arguments**,
- et un **retour chariot** qui accepte la ligne entrée.

2.1. Syntaxe des commandes

Chaque commande dispose de sa propre syntaxe :

Sans options :

```
ls
```

Avec une option :

```
ls -l
```

Avec plusieurs options :

```
ls -l -a -h -t
ls -laht
```

2.2. Options et arguments

Options “double dash” :

```
ls --all
ls --help
```

Donner un argument :

```
ls -l /home
```

Donner plusieurs arguments :

```
ls -l /home /var
```

2.3. Commandes hors du PATH

A titre d’exemple la commande “`ls`” s’exécute car elle est située dans un des chemins indiqués dans la **variable d’environnment PATH**. On peut afficher ces chemins par défaut pour les fichiers exécutables via cette commande :

```
echo $PATH
```

peut afficher ce résultat :

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

On peut exécuter le logiciel directement à partir de l’emplacement absolu :

```
/bin/ls
```

On peut exécuter le logiciel directement à partir de l'emplacement relatif :

```
cd /bin
./ls -l ls
```

2.4. Entrer des commandes

On peut entrer des commandes sur plusieurs lignes :

```
ls /var
ls /home
ls /usr
```

Ou on peut entrer des commandes sur une seule ligne on peut séparer les commandes par un "semicolon", point-virgule, " ; " :

```
ls /var; ls /home; ls /usr
```

2.5. Séquences de commandes

Si les arguments diffèrent pour une même commande, on peut créer une boucle et profiter de variables :

```
for arg in /home /var /usr
do
  echo "visualisation : " $arg
  ls -a $arg
done
```

Ou encore en une seule ligne

```
for arg in /home /usr /var; do ls -la $arg; done
```

Mais cela était inutile :

```
ls /home /var /usr
```

2.6. Codes de retour

La variable spéciale `$?` contient le code de retour d'exécution de la commande précédente.

Une commande qui s'exécute avec succès rend un retour valorisé à `0`. Par exemple la commande `true` rend toujours ce code de retour :

```
true ; echo $?
```

Une commande qui s'échoue rend un retour valorisé à `1`. Par exemple la commande `false` rend toujours ce code de retour :

```
false ; echo $?
```

2.7. Exécutions conditionnelles de base

`&&` et `||` sont des séparateurs de commandes conditionnels.

Opérateur `&&`

Voyez-vous même :

```
true && echo la commande précédente a réussi
```

Ecrit le message “la commande précédente a réussi” si la première commande est exécutée sans erreur.

Par exemple sous Debian/Ubuntu :

```
sudo apt-get update && sudo apt-get -y upgrade
```

Par exemple sous Centos :

```
sudo yum -y install epel-release && sudo yum repolist
```

Opérateur `||`

Voyez-vous même :

```
false || echo la commande précédente a échoué
```

```
commande1 || commande2
```

Ecrit le message “la commande précédente a échoué” si la première commande a échoué.

Par exemple :

```
sudo apt-get update || sudo yum -y update
```

2.7. Historique des commandes

Pour voir la liste des commandes que vous avez validées, vous pouvez utiliser la commande interne de bash `history` :

```
history
```

La commande `history` liste les commandes en cache ainsi que celles sauvées dans `~/.bash_history`. Lorsque l’utilisateur quitte le shell, les commandes en cache sont inscrites dans ce fichier.

Vous pouvez récupérer les commandes tapées en utilisant les flèches directionnelles (haut et bas) de votre clavier.

La commande `history -c` efface l’historique de la session courante.

Historique : raccourcis emacs

Vous pouvez également utiliser des raccourcis emacs qui vous permettent d'exécuter et même de modifier ces lignes.

Par exemple `Ctrl-p`/`Ctrl-n` pour ligne précédente/ligne suivante ou `Ctrl-a` / `Ctrl-e` pour début/fin de ligne.

Pour en savoir plus sur Emacs : https ://www.tuteurs.ens.fr/unix/editeurs/emacs.html

Historique : raccourcis bang

Les raccourcis bang reprennent les commandes précédentes, par exemple :

```
ls -a
^ls^ps
!!
history
!2
```

2.8. Tabulation

Selon la distribution la touche de tabulation offre des possibilités d'auto-complétion.

L'auto-complétion de certains binaires non natifs demandent l'installation et la configuration de scripts ou paquets supplémentaires.

2.9. Substitution de commandes

La commande `uname` permet de connaître la version du noyau courant. Comment la substituer dans une variable ?

```
uname -a
```

```
Linux c7li 3.10.0-327.4.5.el7.x86_64 #1 SMP Mon Jan 25 22:07:14 UTC 2016 x86_64 x86_64 x86_\
64 GNU/Linux
```

Mise en variable de la sortie de la commande de deux manières :

```
system=$(uname -a)
```

ou

```
system=`uname -a`
```

Vérification.

```
echo $system
```

```
Linux c7li 3.10.0-327.4.5.el7.x86_64 #1 SMP Mon Jan 25 22:07:14 UTC 2016 x86_64 x86_64 x86_\
64 GNU/Linux
```

2.10. Alias

Un alias est une autre manière de substituer des commandes.

Pour obtenir la liste des alias :

```
alias
```

```
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

Créer un alias :

```
alias zozo="ls -a"
alias
```

```
zozo
```

Les alias se chargent dans le profil de connexion de l'utilisateur.

Troisième partie Traitement du texte

Objectifs de certification

Linux Essentials

- *Sujet 3 : Le pouvoir de la ligne de commande*
  - 3.2 Recherche et extraction de données à partir de fichiers (valeur : 3)

RHCSA EX200

- **1. Comprendre et utiliser les outils essentiels**
  - 1.2. Utiliser la redirection des entrées/sorties
  - 1.3. Utiliser des expressions grep et régulières pour analyser du texte
  - 1.7. Créer et éditer des fichiers texte
  - 1.11. Localiser, lire et utiliser la documentation système, notamment les manuels, informations et fichiers dans `/usr/share/doc`

LPIC 1

- *Sujet 103 : Commandes GNU et Unix*
  - 103.2 Traitement de flux de type texte avec des filtres
  - 103.4 Utilisation des flux, des tubes et des redirections
  - 103.7 Recherche dans des fichiers texte avec les expressions rationnelles
  - 103.8 Édition de fichier simple

Introduction

Cette partie intitulée “Traitement du texte” s’intéresse aux différents outils Unix de traitement du texte. Un premier chapitre s’intéresse aux outils natifs avec les tubes et les redirections avec les outils comme cat, tac, head, tail, tee, wc, tr, od, join, cut, sort, paste, diff. Un second chapitre s’intéresse aux expressions rationnelles (regexp) et aux outils grep, sed et awk. Enfin un dernier chapitre est consacré à l’éditeur vi.

3. Outils de base de traitement du texte

Ce chapitre expose les notions et les outils de base pour traiter du texte sous Linux. On s'attachera aux notions de tubes et de redirections d'entrée et de sortie et leur interaction avec différents binaires POSIX comme cat, tac, head, tail, tee, wc, split, od, hexdump, cut, uniq, sort, paste, join, fmt, pr, tr et diff.

1. Redirections et tubes

https ://wiki.bash-hackers.org/syntax/redirection

1.1. Redirections et tubes

Les processus UNIX ouvrent trois descripteurs de fichiers standards (correspondant aux flux standards) qui permettent de traiter les entrées et sorties. Ces descripteurs standards peuvent être redéfinis pour chaque processus. Dans la plupart des cas, le descripteur stdin (entrée standard) est le clavier, et les deux descripteurs de sortie, stdout (sortie standard) et stderr (l'erreur standard), sont l'écran.

Un processus et ses 3 descripteurs de fichiers STDIN (0), STDOUT (1) et STERR (2)

```
STDIN < ------ PROCESSUS ---- >
                  |     ---- >> STDOUT
                  |     ---- |
                  |
                  2>
                STDERR
```

1.2. Redirection de l'entrée standard

```
programme < fichier
```

Dans ce cas, les données vont de droite à gauche. L'opérateur "<" ne peut être utilisé qu'avec stdin : on ne peut pas l'utiliser avec les flux de sortie.

Si le fichier contient les instructions l et q (une instruction par ligne), alors dans l'exemple suivant fdisk affichera la table des partitions de `/dev/sda`, puis affichera l'aide puis quittera :

```
cat > fdisk.txt
l
q


[CRTL-D]
```

```
su
fdisk /dev/?da < fdisk.txt
exit
```

Redirection de l'entrée standard :

```
PROCESSUS ---- < ---- FICHIER / PÉRIPHÉRIQUE
         ---- 0< ----
```

1.3. Etiquettes

Une étiquette permet de limiter la redirection. C'est utile par exemple pour donner des arguments à une commande sur plusieurs lignes. Un autre usage est la création de fichiers à partir d'un script. Dans l'exemple suivant, on envoie un émail sur plusieurs lignes avec la commande `mail`.

```
mail mon@adresse <<FIN
ceci
est
un
test
FIN
```

Exercice : Créer des fichiers avec un script bash.

Par exemple un script qui crée un fichier personnalisé :

```
#!/bin/bash
n=1
touch fichier-$n.txt
cat << EOF > fichier-$n.txt
Ceci est le fichier n°$n
Ligne 2
Ligne 3
Ligne 4
EOF
echo "fichier-$n créé"
```

1.4. Redirection de la sortie standard

Les données vont de gauche à droite.

```
programme > fichier
```

Par exemple, avec les droits de **root** :

```
fdisk -l /dev/?da > partitions.txt
```

Ceci lance `fdisk` et redirige la sortie vers le fichier partitions.txt. La sortie n'est pas visible a l'écran. Notez que le shell lit cette commande à partir de la droite : le fichier partitions.txt est d'abord créé s'il n'existait pas auparavant, écrasé dans le cas contraire car l'opérateur "`>`" est utilisé.

L'opérateur "`>>`" ajoute la sortie standard à un fichier sans l'écraser.

Redirection de la sortie standard :

```
          ---- > ----
PROCESSUS ---- >> ---- FICHIER / PÉRIPHÉRIQUE
          ---- 1> ----
```

1.5. Exemples de redirection de la sortie standard

- > crée un nouveau fichier avec la sortie standard
- >> ajoute la sortie au fichier

Par exemple :

```
date > date.txt
cat date.txt
```

```
dim fév 21 04:52:01 CET 2016
```

```
date >> date.txt
cat date.txt
```

```
dim fév 21 04:52:01 CET 2016
dim fév 21 04:53:09 CET 2016
```

```
date > date.txt
cat date.txt
```

```
dim fév 21 04:53:32 CET 2016
```

1.6. Redirection de la sortie erreur standard

```
programme 2> fichier_erreur
```

`stdin`, `stdout` et `stderr` sont représentés respectivement par 0, 1 et 2. Cela nous permet de choisir le flux d'erreur standard. Par exemple, vers une corbeille :

```
ls /fake / 2> /dev/null
```

Par exemple, vers un fichier :

```
ls /fake 2> err.txt
```

Redirection de l'erreur standard :

```
PROCESSUS ---- 2> ---- FICHIER / PÉRIPHÉRIQUE
```

1.7. Travailler avec les redirections

La commande suivante donne des erreurs et une sortie standard :

```
find /etc/ -name "*.crt"
```

```
/etc/ssl/certs/ca-certificates.crt
find: /etc/ssl/private: Permission denied
...
```

Isoler la sortie erreur :

```
find /etc/ -name "*.crt" > /dev/null
```

```
find: /etc/ssl/private: Permission denied
```

Isoler la sortie standard :

```
find /etc/ -name "*.crt" 2> /dev/null
```

```
/etc/ssl/certs/ca-certificates.crt
```

Diviser les sorties :

```
find /etc/ -name "*.crt" 2> /dev/null
/etc/ssl/certs/ca-certificates.crt
```

```
find /etc/ -name "*.crt" > crt.txt 2> crt.err
cat crt.txt
```

```
/etc/ssl/certs/ca-certificates.crt
```

```
cat crt.err
```

```
find: /etc/ssl/private: Permission denied
```

Rediriger une sortie vers l'autre, ici la sortie d'erreur sur la sortie standard :

```
find /etc/ -name "*.crt" 2>&1
```

Le symbôle `&1` identifie la sortie standard demanière certaine. Sans le `&` la sortie d'erreur sera redirigée dans un fichier nommé `1`.

Exemple final récapitulatif à méditer :

Avec le fichier `fdisk.txt`.

```
fdisk /dev/?da < fdisk.txt 2> /dev/null > resultat.txt
```

Le fichier `fdisk.txt` envoie des commandes en entrée à l'exécutable `fdisk`, le résultat sans les erreurs est écrit dans le fichier `resultat.txt`.

1.8. Tubes

```
programme1 | programme2
```

Les tubes ("*pipe*" en anglais) sont représentés par l'opérateur "`|`". Les données vont de gauche à droite. La figure suivante indique comment la sortie standard du premier processus est redirigée vers l'entrée standard du second processus.

Redirection à partir d'un tube :

```
PROCESSUS1 (stdout) ---- | ---- (stdin) PROCESSUS2
```

Exemple : la sortie standard de la première commande devient l'entrée de la seconde commande.

```
ps aux | grep login
```

Note : L'utilitaire `pgrep` fournit le même résultat.

2. Outils de traitement du texte

2.1. `cat` : éditeur rudimentaire

La commande `cat` peut être utilisée comme un éditeur de texte rudimentaire.

```
cat > texte.txt
ligne 1
ligne 2
ligne 3


[Crtl+D]
```

Vous noterez l'utilisation de `Ctrl+D`. Cette commande est utilisée pour clore la saisie.

2.2. `cat` lecteur de texte

On utilise plus couramment `cat` pour envoyer du texte vers la sortie standard.

Les options les plus courantes sont :

- `-n` numéroter chaque ligne de la sortie
- `-b` numéroter uniquement les lignes non vides
- `-A` afficher le retour charriot

Exemples :

```
cat texte.txt
```

```
ligne 1
ligne 2
ligne 3
```

```
cat -n /etc/resolv.conf
```

En fait, la commande `cat` concatène en sortie standard plusieurs fichiers mis en arguments de la commande.

2.3. `tac` lecteur inverse

`tac` fait la même chose que `cat` à l'exception qu'elle lit de la dernière ligne à la première.

```
tac texte.txt
```

```
ligne 3
ligne 2
ligne 1
```

2.4. `head` et `tail`

On utilise souvent les commandes `head` et `tail` pour analyser les fichiers de journaux. Par défaut, ces commandes affichent 10 lignes. En voici les utilisations les plus courantes :

Afficher les 20 premières lignes de `/var/log/messages` :

```
head -n 20 /var/log/messages
head -20 /var/log/messages
```

Afficher les 20 dernières lignes de `/etc/aliases` :

```
tail -20 /etc/aliases
```

`tail` a une option supplémentaire qui nous permet d'afficher la fin d'un texte en commençant par une ligne donnée.

Afficher le texte en partant de la ligne 25 de `/var/log/messages`

```
tail -n +25 /var/log/messages
```

`tail` peut afficher un fichier en continu avec **l'option `-f`**. C'est très pratique pour suivre les modifications d'un fichier en temps réel.

2.5. Commande `tee`

La commande tee permet à la fois de lire un flux et de le rediriger.

Par exemple, `tee` donne la sortie et l'écrit dans le fichier ls1.txt :

```
ls | tee ls1.txt
```

La sortie de la liste de fichiers dont on compte les lignes est redirigée vers la sortie standard et dans le fichier count.txt

```
ls -l *.txt | wc -l | tee count.txt
```

3. Manipulation de texte

- Compter des lignes, des mots, des octets
- Remplacer des tabulations par des espaces
- Afficher les fichiers binaires
- Découper les fichiers
- Sélectionner les champs et les caractères avec cut
- Trouver des doublons
- Trier la sortie
- Couper des fichiers
- Jointure de texte
- Mise en forme de la sortie avec fmt et pr
- Convertir les caractères

3.1. Compter lignes, mots et octets avec la commande `wc`

La commande `wc` compte le nombre d'octets, de mots et de lignes dans les fichiers.

Les options suivantes vous permettent de sélectionner ce qui nous intéresse :

- `-l` compte le nombre de lignes
- `-w` compte le nombre de mots (words)
- `-c` compte le nombre d'octets
- `-m` compte le nombre de caractères
- sans argument, wc compte ce qui est saisi dans stdin.

Par exemple :

```
wc -l /etc/passwd
cat /etc/passwd | wc -l
```

3.2. Remplacer les tabulations par des espaces

On utilise la commande `expand` pour remplacer les tabulations par des espaces.

`unexpand` est utilisé pour l'opération inverse.

3.3. Afficher les fichiers binaires

Il y a nombre d'outils pour cela. Les plus courants sont `od` (octal dump) et `hexdump`.

3.4. Découper les fichiers avec la commande `split`

La commande `split` peut découper un fichier en plusieurs fichiers plus petits à partir de critères comme la taille ou le nombre de lignes. Par exemple, nous pouvons découper `/etc/passwd` en fichiers de 5 lignes chacun :

```
split -l 5 /etc/passwd
```

Cette commande va créer des fichiers appelés `xaa`, `xab`, `xac`, `xad`, etc., chaque fichier contenant au plus 5 lignes. Tentez et vérifiez :

```
split -dl 5 /etc/passwd passwd
```

Il est possible de donner un préfixe plus significatif que "`x`", comme "`pass-5`" :

```
split -l 5 /etc/passwd passwd-5
```

Cette commande crée des fichiers identiques à la commande précédente, mais ils sont désormais nommés `passwd-5aa`, `passwd-5ab`, `passwd-5ac`, `passwd-5ad`, …

3.5. Sélectionner les champs et les caractères avec `cut`

La commande `cut` peut extraire une plage de caractères ou de champs de chaque ligne d'un texte.

- L'option `-c` est utilisée pour manipuler les caractères.
- Syntaxe : `cut –c {plage1,plage2}`

Exemple :

```
cut -c5-10,15- /etc/passwd
```

Cette commande extrait les caractères 5 à 10 puis 15 jusqu'à la fin pour chaque ligne de `/etc/passwd`.

On peut spécifier le séparateur de champ (espace, virgule, etc.) d'un fichier ainsi que les champs à extraire. Ces options sont définies respectivement par les options `-d` (delimiter) et `-f` (field).

Syntaxe :

- `cut -d {séparateur} -f {champs}`

Exemple :

```
cut -d: -f 1,7 --output-delimiter=" " /etc/passwd
```

Cette commande extrait les 1er et 7e champs de `/etc/passwd` séparés par un espace. Le délimiteur de sortie est le même que le délimiteur d'entrée d'origine (par défaut, la tabulation). L'option `--output-delimiter` vous permet de le changer.

3.6. Trouver des doublons avec la commande `uniq`

Éliminer les lignes successives en doublon : La commande `uniq` n'envoie à STDOUT qu'une version des lignes successives identiques. Par exemple :

```
uniq > /tmp/list1
ligne 1
ligne 2
ligne 2
ligne 3
ligne 3
ligne 3
ligne 1
^D


cat /tmp/UNIQUE
sort | uniq > /tmp/UNIQUE
```

3.7. Trier la sortie avec la commande `sort`

Par défaut, `sort` trie le texte par ordre alphabétique. Pour effectuer un tri numérique, utilisez l'option `-n`.

```
cat << EOF > /tmp/list2
ligne 1
ligne 2
ligne 2
ligne 1
ligne 3
ligne 2
ligne 3
ligne 1
EOF


sort /tmp/list2
sort /tmp/list2 | uniq > /tmp/list3
```

3.8. Jointure de texte avec `paste`

La commande la plus facile est `paste` qui "concatène" deux fichiers l'un à la suite de l'autre.

Syntaxe :

```
paste texte1 texte2
```

Exemples avec deux fichiers :

texte1 :

```
cat << EOF > texte1
01 Paris
02 Luxembourg
03 Berlin
04 Bruxelles
05 Londres
EOF
```

texte2 :

```
cat << EOF > texte2
01 France
02 Grand-Duché de Luxembourg
03 Allemagne
04 Belgique
05 Royaume-Uni
EOF
```

```
paste texte1 texte2
```

```
01 Paris        01 France
02 Luxembourg        02 Grand-Duché de Luxembourg
03 Berlin        03 Allemagne
04 Bruxelles        04 Belgique
05 Londres        05 Royaume-Uni
```

```
paste -s texte1 texte2
```

```
01 Paris        02 Luxembourg        03 Berlin        04 Bruxelles        05 Londres
01 France        02 Grand-Duché de Luxembourg        03 Allemagne        04 Belgique        05 Royaume-Uni
```

```
paste -s -d: texte1 texte2
```

```
01 Paris:02 Luxembourg:03 Berlin:04 Bruxelles:05 Londres
01 France:02 Grand-Duché de Luxembourg:03 Allemagne:04 Belgique:05 Royaume-Uni
```

```
paste -d: texte1 texte2
```

```
01 Paris:01 France
02 Luxembourg:02 Grand-Duché de Luxembourg
03 Berlin:03 Allemagne
04 Bruxelles:04 Belgique
05 Londres:05 Royaume-Uni
```

3.9. Jointure de texte avec `join`

Avec `join` vous pouvez en plus préciser quels champs vous souhaitez à condition que les fichiers disposent d'un début de ligne commun.

Syntaxe :

```
join -j1 {champ_no} -j2{champ_no} texte1 texte2
```

ou

```
join -1 {champ_no} -2{champ_no} texte1 texte2
```

Le texte n'est envoyé à la sortie que si les champs sélectionnés correspondent.

Les comparaisons se font ligne par ligne et le processus s'arrête dès qu'il n'y a pas de correspondance, même s'il y a d'autres correspondances à la fin du fichier.

Par exemple avec les fichiers précédents :

```
join texte1 texte2
```

```
01 Paris France
02 Luxembourg Grand-Duché de Luxembourg
03 Berlin Allemagne
04 Bruxelles Belgique
05 Londres Royaume-Uni
```

Exercice optionnel : Regroupez les fichiers séparés précédemment.

3.10. Mise en forme de la sortie avec `fmt` et `pr`

Vous pouvez modifier le nombre de caractères par ligne avec `fmt`. Par défaut `fmt` joint les lignes et génère des lignes de 75 caractères.

Options de `fmt` :

- `-w` (width) nombre de caractères par ligne
- `-s` découpe les lignes longues mais sans les remplir
- `-u` sépare chaque mot par une espace et chaque phrase par deux espaces
- On peut paginer les longs fichiers pour qu'ils correspondent à une taille donnée avec la commande `pr`. On peut contrôler la longueur des pages (66 lignes par défaut), la largeur (par défaut 72 caractères) ainsi que le nombre de colonnes.
- Lorsqu'on produit un texte sur plusieurs colonnes, chaque colonne est tronquée uniformément en fonction de la largeur de page spécifiée. Cela veut dire que des caractères sont supprimés à moins d'avoir édité le texte de façon à éviter cela.

Par exemple :

```
curl -s https://lipsum.com/feed/json -o lipsum.txt
fmt < lipsum.txt
```

3.11. Convertir les caractères avec la commande `tr`

La commande `tr` convertit un ensemble de caractères en un autre.

Convertir les majuscules en minuscules :

```
tr 'A-B' 'a-b' < fichier.txt
```

Changer de délimiteur dans `/etc/passwd`

```
tr ':' ' ' < /etc/passwd
```

```
join texte1 texte2 | tr ' ' ':'
```

```
01:Paris:France
02:Luxembourg:Grand-Duché:de:Luxembourg
03:Berlin:Allemagne
04:Bruxelles:Belgique
05:Londres:Royaume-Uni
```

Remarque : `tr` a seulement deux arguments ! Le fichier n'est pas un argument.

3.12. Différentiel avec la commande diff

```
diff fichier1 fichier2
```

Quatrième partie Arborescence de fichiers

Objectifs de certification

Linux Essentials

- *Sujet 2 : Trouver son chemin sur un système Linux*
  - 2.3 Utilisation des répertoires et liste des fichiers (valeur : 2)
  - 2.4 Création, déplacement et suppression de fichiers (valeur : 2)
- *Sujet 3 : Le pouvoir de la ligne de commande*
  - 3.1 Archivage de fichiers en ligne de commande (valeur : 2)

RHCSA EX200

- **1. Comprendre et utiliser les outils essentiels**
  - 1.6. Archiver, compresser, décompacter et décompresser des fichiers, à l'aide de `tar`, `star`, `gzip` et `bzip2`
  - 1.8. Créer, supprimer, copier et déplacer des fichiers et des répertoires
  - 1.9. Créer des liens physiques et symboliques
  - 1.11. Localiser, lire et utiliser la documentation système, notamment les manuels, informations et fichiers dans `/usr/share/doc`

LPIC 1

- *Sujet 102 : Installation de Linux et gestion de paquetages*
  - 102.1 Conception du schéma de partitionnement
- *Sujet 103 : Commandes GNU et Unix*
  - 103.3 Gestion élémentaire des fichiers
- *Sujet 104 : Disques, systèmes de fichiers Linux , arborescence de fichiers standard (FHS)*
  - 104.6 Création et modification des liens physiques et symboliques sur les fichiers
  - 104.7 Recherche de fichiers et placement des fichiers aux endroits adéquats

Introduction

Cette partie intitulée "Arborescence de fichiers" permet dans un premier temps d'appréhender la structure d'un système de fichier Linux en s'y déplaçant. Un second chapitre traite des opérations sur les fichiers (inodes, listing, création de fichiers, de répertoires, déplacements, copies, effacement, liens). Un troisième chapitre se consacre entièrement à la pratique de recherche de fichier sur un système Linux. Enfin un dernier chapitre traitre des outils d'archivage et de compression de fichiers comme gzip, bzip2, tar ou zip.

4. Filesystem Hierachy Standard (FHS)

Ce chapitre se focalise sur la structure du système de fichiers sous Linux. On identifiera les différents emplacements d'un système Linux, l'emplacement courant et comment naviguer à travers des chemins relatifs ou absolus.

https ://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

1. La structure du système de fichier

Un système de fichiers est similaire à une arborescence, avec une racine qui se scinde en branches et sous-branches, soit en répertoires et sous-répertoires.

On commence par le tronc principal, la racine (root) : `/`. C'est un peu comme le `C :\` sous DOS, sauf que `C :\` est également le premier périphérique de stockage, alors que la racine peut correspondre à n'importe quel disque (partition) de votre système (point de montage).

La racine contient différents répertoires et sous-répertoires contenant eux-mêmes des fichiers.

2. La commande `tree`

La commande `tree` liste le contenu de répertoires sous forme d'arborescence.

```
sudo yum install -y tree
```

Par exemple sous Centos 7, l'arborescence à partir de la racine :

```
tree -L 1 /
```

```
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
```

```
├── usr
└── var

20 directories, 0 files
```

2. Partition racine

Les répertoires suivants peuvent être montés sur d'autres partitions que la celle de la racine :

- `/boot`
- `/home`
- `/root`
- `/tmp`
- `/usr`
- `/usr/local`
- `/opt`
- `/var`

Les répertoires `/dev`, `/bin`, `/sbin`, `/etc` et `/lib` **doivent être montés sur la partition racine**.

De plus, la racine doit contenir un répertoire `/proc` vide. Il est utilisé par le noyau pour informer sur le statut du système d'exploitation (processus, statistiques d'utilisation de la mémoire, etc.).

3. Contenu du système de fichier

- `/bin` et `/sbin` : contiennent les binaires nécessaires au démarrage et les commandes essentielles
- `/dev` : fichiers périphériques ou fichiers spécifiques
- `/etc` : fichiers et répertoires de configuration spécifiques à la machine
- `/lib` et `/lib64` : bibliothèques partagées pour les binaires de `/bin` et `/sbin`. Contient également les modules du noyau.
- `/mnt` ou `/media` : points de montage pour les systèmes de fichiers externes
- `/proc` : informations du noyau. En lecture seule sauf pour `/proc/sys`.
- `/boot` : contient le noyau Linux, le System.map (carte des symboles du noyau) et les chargeurs d'amorçage secondaires.
- `/home` (facultatif) : répertoires utilisateurs, avec, en général, une copie du contenu de `/etc/skel`.
- `/root` : répertoire de l'utilisateur root.
- `/sys` : export d'information du noyau, à la manière de `/proc`.
- `/tmp` : fichiers temporaires.
- `/usr` : User Specific Ressource. Contenu essentiellement statique et partageable. `/usr` est composé de sous-répertoires `bin`, `sbin`, `lib` et autres qui contiennent des programmes et bibliothèques de votre système non essentielles ni nécessaires au démarrage.
- `lost+found` : est un dossier spécial de récupération des données du système de fichiers.

```
$ tree -L 1 /usr
/usr
├── bin
├── etc
├── games
├── include
├── lib
├── lib64
├── libexec
├── local
├── sbin
├── share
├── src
└── tmp -> ../var/tmp

12 directories, 0 files
```

- `/usr/local` ou `/opt` : programmes et bibliothèques supplémentaires. En général, c'est dans ces répertoires que l'on place les programmes qui ne font pas partie des paquets des distributions.
- `/var` : données variables comme les spool ou les journaux. Les sous-répertoires peuvent être soit partageables (comme `/var/spool/mail`) soit non partageables (comme `/var/log`).
- `/var/www`, `/var/ftp` ou `/srv` : pages web ou fichiers ftp anonymes.

4. Chemins relatifs et absolus

On peut accéder à un répertoire ou un fichier en donnant son chemin complet, qui commence à la racine (`/`), ou en donnant son chemin relatif partant du répertoire courant.

Chemin absolu :

- indépendant du répertoire de travail de l'utilisateur
- commence par `/`

Chemin relatif :

- dépend de l'endroit où se trouve l'utilisateur
- ne commence pas par `/`

5. Se déplacer dans le système de fichiers

Comme pour tout système de fichiers structuré, un certain nombre d'outils aident à parcourir le système. Les deux commandes suivantes sont des commandes internes du shell :

- `pwd` : (Print Working Directory) affiche le répertoire actuel en chemin absolu
- `cd` : la commande pour changer de répertoire (Change Directory)

6. Emplacements

- L'emplacement courant est représenté par un point `.`
- L'emplacement parent est représenté par deux points `..`
- Le répertoire utilisateur courant est représenté par le tild `~`

7. Exercices

- Aller dans le répertoire `/etc/`
- Revenir dans le répertoire personnel Téléchargements
- Aller dans `/etc/default` de manière relative
- Revenir rapidement dans son répertoire personnel
- Se placer dans le répertoire `Téléchargements`

Cinquième partie Sécurité locale

Objectifs de certification

Linux Essentials

- *Sujet 5 : Sécurité et droits d'accès aux fichiers*
  - 5.1 Sécurité élémentaire et identification des catégories d'utilisateurs (valeur : 2)
  - 5.2 Création des utilisateurs et des groupes (valeur : 2)
  - 5.3 Gestion des propriétés et des droits d'accès aux fichiers (valeur : 2)
  - 5.4 Répertoires et fichiers spéciaux (valeur : 1)

RHCSA EX200

- **1. Comprendre et utiliser les outils essentiels**
  - 1.5. Se connecter et changer d'utilisateur dans des cibles à plusieurs utilisateurs
  - 1.10. Répertorier, définir et modifier des autorisations `ugo`/`rwx` standard
  - 1.11. Localiser, lire et utiliser la documentation système, notamment les manuels, informations et fichiers dans `/usr/share/doc`
- **8. Gérer des groupes et utilisateurs système**
  - 8.1. Créer, supprimer et modifier des comptes utilisateur locaux
  - 8.2. Modifier les mots de passe et ajuster la durée de validité des mots de passe pour les comptes utilisateur locaux
  - 8.3. Créer, supprimer et modifier des groupes locaux et des appartenances de groupe
  - 8.4. Configurer l'accès super-utilisateur
- **9. Gérer la sécurité**
  - 9.2. Créer et utiliser des access control lists sur les fichiers
- **5. Créer et configurer des systèmes de fichiers**
  - 5.4. Créer et configurer des répertoires Set-GID pour la collaboration
  - 5.8. Détecter et résoudre les problèmes d'autorisation sur les fichiers

LPIC 1

- *Sujet 104 : Disques, systèmes de fichiers Linux , arborescence de fichiers standard (FHS)*
  - 104.5 Gestion des permissions et de la propriété sur les fichiers
- *Sujet 107 : Tâches d'administration*
  - 107.1 Gestion des comptes utilisateurs et des groupes ainsi que des fichiers systèmes concernés
- *Sujet 110 : Securité*
  - 110.1 Tâches d'administration de sécurité
  - 110.2 Configuration de la sécurité du système
  - 110.3 Sécurisation des données avec le chiffrement

LPIC2

- *Sujet 206 : Maintenance système*
  - 206.3 Information des utilisateurs (valeur : 1)
- *Sujet 210 : Gestion des clients réseau*
  - 210.2 Authentification PAM (valeur : 3)

Introduction

Cette partie intitulée “Sécurité locale” s’intéresse aux utilisateurs et groupes Linux ainsi que tous les éléments qui concernent leur gestion : mots de passe, connexions su et sudo, opérations sur les utilisateurs et groupes (création, ajout, désactivation, effacement), les permissions (rwx, ugo, SUID, SGID, Sticjy bit), les ACLs sur les fichiers et enfin l’usage de PAM.

5. Utilisateurs et groupes Linux

Dans ce chapitre sur les utilisateurs et les groupes, nous verrons la différence entre les programmes su et sudo. On identifiera l'emplacement des informations sur les utilisateurs et les groupes. On trouvera aussi des considérations sur le chiffrement et la force des mots de passe sous Linux.

1. Commande `su`

`su` (substitute user ou switch user) est une commande Unix permettant d'exécuter un interpréteur de commandes en changeant d'identifiant de GID et de UID.

> Sans argument, la commande utilise les UID `0` et le GID `0`, c'est-à-dire ceux du compte utilisateur root.

Cette commande est surtout utilisée pour obtenir les privilèges d'administration à partir d'une session d'utilisateur normal, c'est-à-dire, non privilégiée.

> L'option `-` place le shell de l'utilisateur.

Exercice

```
su
exit
```

```
su -
exit
```

```
su root
exit
```

```
su - root
exit
```

2. Programme `sudo`

`sudo` (abréviation de substitute user do, en anglais : « exécuter en se substituant à l'utilisateur ») est une commande qui permet à l'administrateur système d'accorder à certains utilisateurs (ou groupes d'utilisateurs) la possibilité de lancer une commande en tant qu'administrateur, ou comme autre utilisateur, tout en conservant une trace des commandes saisies et des arguments.

Exercice

Pour configurer sudo, dans une session utilisateur `root` :

visudo

Visudo

La commande visudo ouvre le fichier de configuration sudo avec l'éditeur vi. En voici le contenu sous Centos 7 :

```
# Adding HOME to env_keep may enable a user to run unrestricted
# commands via sudo.
#
# Defaults   env_keep += "HOME"

Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##         user        MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root        ALL=(ALL)         ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel        ALL=(ALL)         ALL

## Same thing without a password
# %wheel        ALL=(ALL)         NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
```

Un utilisateur devient "sudoer" en modifiant finement ce fichier ou ses fichiers inclus jusqu'à définir les commandes seules autorisées. On peut profiter comme dans cet exemple d'un configuration par défaut qui rend "sudoer" tout membre d'un groupe défini (le groupe wheel sous Centos et le groupe sudo sous Debian / Ubuntu).

Par exemple, ajouter un utilisateur au système en tant que non-root :

```
sudo useradd zozo
```

Exercice : se configurer en tant qu'utilisateur normal comme "sudoer".

3. Utilisateurs

Toute entité (personne physique ou programme particulier) devant interagir avec un système UNIX est authentifiée sur cet ordinateur par un utilisateur ou "*user*".

Ceci permet d'identifier un acteur sur un système UNIX. Un utilisateur est reconnu par un nom unique et un numéro unique.

Sur tout système UNIX, il y a un **super-utilisateur**, généralement appelé *root*, qui a tous les pouvoirs sur le système. Il peut accéder librement à toutes les ressources de l'ordinateur, y compris à la place d'un autre utilisateur, c'est-à-dire sous son identité. En général, du moins sur les systèmes de production, seul l'administrateur système possède le mot de passe root. L'utilisateur root porte le numéro 0.

4. Utilisateurs : fichier `/etc/passwd`

On peut créer un utilisateur de plusieurs manières mais la finalité est toujours la même : pour chaque utilisateur, une entrée doit être créée dans le fichier `/etc/passwd` sous ce format :

```
account:passwd:UID:GID:GECOS:directory:shell
```

Illustration

Par exemple, on ajoute un utilisateur "user1" :

```
echo "user1:x:2000:2000:user1:/home/user1:/bin/bash" >> /etc/passwd
```

Mais faut-il encore créer le groupe correspondant, vérifier la validité des UID et GID, créer le répertoire utilisateurs, y donner les droits et y placer une structure ...

5. Mots de passe : fichier `/etc/shadow`

Le mot de passe est écrit dans le fichier `/etc/shadow` avec ses paramètres :

1. nom de connexion de l'utilisateur (« login »)
   - mot de passe chiffré : `$1$` (MD5),`$2$` (Blowfish), `$5$` (SHA-256), `$6$` (SHA-512)
   - date du dernier changement de mot de passe
   - âge minimum du mot de passe
   - âge maximum du mot de passe
   - période d'avertissement d'expiration du mot de passe
   - période d'inactivité du mot de passe
   - date de fin de validité du compte
   - champ réservé

Illustration

```
francois:$6$d/uLirbD$s90XRAj6g14036jIuvYYQaSOSrcJKqiNNywIQplztkTlyIrySZE1o2zjFvSobewvyORXFd\
Z7bGeF0U1OTPoOm.:16842:0:99999:7:::
```

6. Générer un mot de passe aléatoire

Exercice

`pwmake` est un outil qui permet de générer des mots de passe (Centos 7) :

```
pwmake 128
```

```
Ib9AHK3boravZUSuNuffYPExunEn
```

Exercice

Voici un exemple à utiliser dans un exercice de récupération de mot de passe :

```
pwmake 128 | passwd --stdin root
```

```
Changing password for user root.
passwd: all authentication tokens updated successfully.
```

Exercice

On peut utiliser des outils natifs.

Avec les utilitaires de génération d'empreinte :

```
date +%s | sha256sum | base64 | head -c 32 ; echo
```

Avec `/dev/urandom` :

```
< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-32};echo;
```

Avec openssl s'il est installé :

```
openssl rand -base64 32
```

Dans les dépôts Debian, on trouve les générateurs de mots de passe :

- `pwgen`
- `apg`
- `makepasswd`

7. Tester la force des mots de passe

On peut tester la force des mots de passe avec *John The Ripper*.

Si le paquet `john` est présent dans le dépôt Debian / Ubuntu, il n'est pas disponible pour les distributions RHEL. On peut alors le compiler soi-même :

```
#!/bin/bash
# Centos 7/8 John the Ripper Installation
#release=(j 1.8.0)
release=(k 1.9.0)
# Check Centos version
if [ -f /etc/redhat-release ] ; then
  source /etc/os-release
  if [ $VERSION_ID == "8" ] ; then
    packager=dnf
  elif [ $VERSION_ID == "7" ] ; then
    packager=yum
  fi
else exit ; fi
sudo ${packager} -y install wget gpgme
sudo ${packager} -y group install "Development Tools"
cd
wget http://www.openwall.com/john/${release[0]}/john-${release[1]}.tar.xz
wget http://www.openwall.com/john/${release[0]}/john-${release[1]}.tar.xz.sign
wget http://www.openwall.com/signatures/openwall-signatures.asc
gpg --import openwall-signatures.asc
gpg --verify john-${release[1]}.tar.xz.sign
tar xvfJ john-${release[1]}.tar.xz
cd john-${release[1]}/src
make clean linux-x86-64
cd ../run/
./john --test
#password dictionary download
wget -O - http://mirrors.kernel.org/openwall/wordlists/all.gz | gunzip -c > openwall.dico
```

et puis :

```
cd john-*/run
./john /etc/shadow
```

```
Loaded 4 password hashes with 4 different salts (generic crypt(3) [?/64])
testtest         (tintin)
testtest         (root)
testtest         (francois)
testtest         (gustave)
guesses: 4  time: 0:00:02:25 DONE (Tue Feb  3 23:06:29 2015)  c/s: 170  trying: spazz - das\
ha
Use the "--show" option to display all of the cracked passwords reliably
```

8. Groupes

Un utilisateur UNIX appartient à un ou plusieurs groupes.

Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs.

Le groupe principal est le groupe initial de l'utilisateur.

L'utilisateur peut appartenir à des groupes secondaires.

9. Fichiers `/etc/group` et `/etc/gshadow`

Les fichiers `/etc/group` et `/etc/gshadow` définissent les groupes.

Le fichier `/etc/group` comporte `4` champs séparés par " :" :

1. nom du groupe
2. mot de passe du groupe (ou `x` si le fichier `gshadow` existe)
3. le GID
4. liste des membres séparés par une virgule

10. Appartenance à un groupe

On peut vérifier son identifiant et l'appartenance aux groupes via les commandes `id` et `groups` :

La commande

```
id
```

offre par exemple ce résultat :

```
uid=1000(francois) gid=1000(francois) groupes=1000(francois),10(wheel) contexte=unconfined_\
u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

La commande

```
groups
```

donne ceci :

```
francois wheel
```

Révisions

Page vide.