# Etudier et s'autoévaluer en vue d'obtenir le "Linux Essentials Certificate of Achievement"

Document en construction ! (document de travail à usage interne : lpi_linux_essential)

## Définition de la certification

# Cursus LPI Linux Essentials Certificate of Achievement

## Objectifs

# 0 Notions de base en informatique

L'informatique constitue l'ensemble des disciplines scientifiques et techniques s'appliquant au traitement de l'information. L'ordinateur ...

- est une machine électronique;
- est programmable (plusieurs applications sont possibles);
- est destiné à traiter l'information
- accède aux données et instructions stockées dans sa mémoire
- reçoit des informations en entrée (input)
- effectue des actions en sortie (output)
- Son aspect physique, le matériel, les composants (Hardware) conditionnent les performances.
- Les logiciels (Software) diffèrent au gré des applications. Ils gèrent la logique d'une application et ses possibilités de traitement.
- Un logiciel particulier permet le fonctionnement de l'ordinateur, il s'agit du système d'exploitation.

Les périphériques informatiques en entrée/sortie sont par exemple,

- en entrée : un clavier, une souris, un scanner, une webcam, un microphone,...
- en sortie : un écran, une imprimante,...
- en entrée ET sortie : un disque dur, une clé USB, le réseau, un lecteur/graveur de CD/DVD,...

Pour en savoir plus, consulter http://fr.wikipedia.org/wiki/Ordinateur et pages liées.

- Codage de l'information :
  - Système binaire. L'unité de base de ce système est un bit, qui permet de coder 2 valeurs (0/1, vrai/faux, haut/bas, noir/blanc...)
  - Un nombre binaire de 4 chiffres (0 ou 1) permet de coder seize valeurs différentes qui peuvent être représentées par un seul caractère hexadécimal
  - Un nombre binaire de 8 chiffres ou hexadécimal de 2 chiffres permet de coder 256 valeurs différentes. Il est appelé octet ou aussi byte en pratique.

    - Les processeurs sont conçus pour traiter simultanément plusieurs octets (on parle alors de mots). Les plus courants sont des processeurs 32 bits, permettant d'accéder (sans artifice) 4 294 967 296 entités mémoires différentes, ou 64 bits, permettant d'accéder 18 446 744 073 709 551 616 entités mémoires différentes.
    - Le codage des caractères : ISO 8859-15, Unicode, UTF-8
- Aspects historiques et chronologiques de l'informatique
- Aspects matériels :
    - Le processeur ou CPU
    - La mémoire
    - Les disques durs
    - carte mère
    - carte graphique
    - Les écrans
- Les systèmes d'exploitation
- Les logiciels
- Organisation des fichiers et répertoires
- La ligne de commande (Command Line Interface)
- Les interfaces graphiques (gestionnaires de bureau, Graphical User Interface)
- Firmwares
    - BIOS
    - EFI, UEFI
- Les utilisateurs, droits d'exécution et droits d'accès
- Réseaux et Internet
- Sécurité et légalité
- Licences libres - licences propriétaires

### Exercices

- Estimer en ordre de grandeur le nombre de caractères d'une ligne, d'une page de texte, d'un roman, d'un dictionnaire courant, d'un journal, d'une encyclopédie, d'une bibliothèque universitaire, ...
- Conversions entre représentations binaire, décimale ou hexadécimale :
    - quelles sont les écritures décimale et hexadécimale du nombre binaire 10100110
    - quelles sont les écritures décimale et binaire du nombre hexadécimal C7
    - quelles sont les écritures binaire et hexadécimale du nombre décimale 137
- Analyser la configuration matérielle de votre ordinateur
- Quel est l'espace mémoire nécessaire pour mémoriser le contenu d'un écran ou d'une image, sans compression ? (utiliser aussi des définitions des appareils photos, scanners,...)
- Quels sont les différents types de mémoires et les différences essentielles ?

# 1 La communauté Linux et une carrière dans l'Open Source

Pour ce qui concerne les systèmes d'exploitation accessibles au grand public, l'éditeur de logiciel Microsoft propose son OS propriétaire, Windows, qui domine en terme de parts de marché. A côté, le constructeur d'ordinateurs Apple propose son OS, OS X, sur ses Mac et ses systèmes nomades.

En dehors de ces systèmes propriétaires, Linux est un OS sous licence libre, absolument indépendant des constructeurs, et fonctionnant sur la plupart des matériels. Une communauté assez complexe

allant des développeurs aux utilisateurs s'est crée autour de ce projet libre, gérant un écosystème particulièrement complexe d'applications, des plus simples aux très complexes. Ces projets peuvent être élaborés dans un cadre purement communautaire et bénévole, ou dans des entités fonctionnant sur des principes d'entreprises, incluant des aspects commerciaux, et par conséquent des perspectives de carrières professionnelles.

# 1.1 Evolution de Linux et des distributions populaires

La naissance de Linux date de 1991, grâce à l'impulsion de son créateur, Linus Torvalds, qui a été très vite aidé par de nombreux volontaires séduits par le mode de développement communautaire de ce noyau de système d'exploitation respectant les fonctionnalités de UNIX et par sa licence de distribution libre, la GNU GPL. Pour constituer véritablement un système d'exploitation complet, il fallait adjoindre d'autres éléments à ce noyau, qui était précisément la pièce manquante du projet GNU, porté par Richard Stallman depuis 1983, qui comportait déjà les autres éléments essentiels : le compilateur GCC, l'éditeur de texte Emacs, un interface en ligne de commande (shell), des bibliothèques logicielles, les commandes Unix, des outils réseau, le serveur graphique X ... finalement, en réunissant ces deux projets, le système d'exploitation libre GNU/Linux était né, et il entamait alors une formidable progression au milieu des années nonantes.

Dès cette époque, Linux est devenu un OS principalement utilisé pour des serveurs. L'utilisation sur le poste utilisateur était limitée à une catégorie d'utilisateurs férus d'informatique, de geeks.

Plus tard, le développement d'environnements de bureau suffisamment matures a simplifié l'accès à Linux pour de nombreux “profanes”, l'administration et les réglages du système devenant accessibles par des interfaces utilisateurs graphiques. Dès les années 2005, il devenait clair que Linux pouvait satisfaire les besoins d'un utilisateur lambda. Nombre d'entre eux n'ont pas hésité à faire le pas, déçus par le système d'exploitation qui s'était imposé lors de l'achat de leur ordinateur.

A l'aube des années 2010 Linux a encore franchi une étape supplémentaire dans la popularité, en prenant au travers du système Android une place de leader sur le marché des smartphones et des tablettes. Connaissant aussi ses nombreuses utilisations dans des appareillages spécifiques : routeurs, décodeurs, télévisions, GPS, mediacenter, on peut à présent affirmer que Linux est présent de manière significative dans chaque foyer.

Pour celui qui voudra installer, utiliser et comprendre un système Linux complet, le choix d'une distribution est un passage obligé. Les distributions proposent non seulement un environnement complet, mais aussi les moyens de l'étendre par l'ajout de programmes, ou de la mettre à jour. Des communautés d'utilisateurs et de développeurs lui sont souvent associées, ainsi que des sociétés prestataires de services commerciaux.

Toutes semblables, toutes différentes, elles se distinguent entre elles par divers critères, dont ceux-ci :

- l'environnement de bureau par défaut (GNOME, KDE, LXDE, XFCE,...)
- un système de gestion de paquets de logiciels
- le nombre de paquets proposés
- Un modèle de développement communautaire, ou reposant sur une société commerciale
- le rythme de sorties des versions
- l'existence de versions plus stables, suivies sur le long terme
- des supports matériels plus ou moins important

Les principales distributions :

- Red Hat
- Debian
- SUSE
- Ubuntu] * [http://fr.wikipedia.org/wiki/Linux_Mint|Linux Mint
- ArchLinux
- Gentoo

Pour un comparatif des distributions, *cf.* le site Distrowatch.

### Exercices

1. Proposer des distributions Linux adaptées aux contraintes suivantes :
    1. Ordinateur ancien, avec peu de mémoire (256 Mo), usages : bureautique, internet, lecture de mp3, classements et traitements élémentaires de photos, vidéos
    2. Ordinateur récent et performant. Usages particuliers : montages vidéos, photographie numérique, jeux (avec les dernières versions des logiciels)
    3. Serveur personnel (fichiers, web), robuste et à oublier, sans trop de mises à jour nécessaires
    4. Mediacenter
    5. Distribution bien finie côté esthétique, avec une grande communauté d'entraide

## 1.2 Principales applications Open Source

Les applications sont innombrables, aussi nous nous limiterons à préciser les types de programmes très souvent présents par défaut dans la plupart des distributions, en indiquant les catégories de classement utilisées.

- Accessoires
    - Éditeur de texte (gedit)
    - Calculatrice
    - Scan (Xsane)
- Bureautique
    - Traitement de texte (writer)
    - Tableur (calc)
    - Logiciel de présentation (presentation)
- Graphisme
    - Traitement d'image (GIMP)
    - Graphisme vectoriel (Inkscape)
    - Animation 3D (Blender)
- Internet
    - Navigateur web (Firefox, Chromium)
    - Client de courrier (Thunderbird)
    - Messagerie instantanée
    - Serveurs (Apache, MySQL, SSH,...)
- Jeux
- Outils systèmes
    - Bureau à distance
    - Compression
    - Logithèque
- Programmation
    - Nombreux compilateurs ou interpréteur (C, C++, java, Python, Perl, Ruby, PHP,...)

- Son et vidéos
  - Éditeur de sons (Audacity)
  - Éditeur vidéo (Cinelerra, openshot)
  - Transcodeur vidéo (Avidemux)
  - Lecteurs son, vidéo
  - Télévision numérique
  - Enregistreur de son

Différents outils permettent d'ajouter des programmes parmi un très large choix. Les téléchargements et l'installation s'opèrent en utilisant des dépôts reconnus.

## 1.3 Comprendre les logiciels Open Source et leurs licences

## 1.4 Compétences ICT (Information & communications technology) et travail sous Linux

# 2 Trouver son chemin sur un système Linux (pondération : 8)

## 2.1 Les bases de la ligne de commande

### 2.1.1 Introduction

Les utilisateurs de Windows qui passent à Linux risquent d'être déstabilisés à la vue de l'interface «ligne de commandes».
Toutefois les versions récentes de Linux intègrent également une interface graphique équivalente à celle utilisée dans Windows, Mac OSX ….

L'objectif est de vous familiariser avec l'interface texte ( ou terminal ) pour utilisateur, connue sous le nom «Shell».
Le « Shell » est un moyen pratique pour réaliser des opérations qui peuvent être difficile à exprimer graphiquement.

### 2.1.2 Le «Shell», c'est quoi?

Les utilisateurs ne peuvent pas communiquer directement avec le noyau du système d'exploitation. En fait cela n'est seulement possible qu'à travers des programmes, comme le «Shell», qui y accède via les «appels systèmes (system call interface)». Le Shell est donc un programme spécial pour utilisateur. Il va lire les commandes introduites au clavier et les interpréter comme commandes a être exécutées. Le Shell agit donc comme «interface» vis à vis de l'ordinateur qui enferme le système d'exploitation un peu comme une coquille et le cache ainsi de l'utilisateur. Le « Shell » est un des nombreux programmes qui accèdent au système d'exploitation.

La courbe d’apprentissage du Shell est plus longue parce que moins intuitive, mais sur certains points comme le traitement par lots de plusieurs fichiers, la ligne de commande conserve un avantage évident.

Types de Shell :

le /bin/sh shell Bourne
le /bin/bash shell Bourne Again Shell
le /bin/csh C shell
le /bin/ksh Korn shell
le /bin/tcsh C shell amélioré

Le principe de base est toujours resté le même : Les shells sont des interpréteurs, ils lisent chaque commande saisie par l'utilisateur, vérifient et traitent la syntaxe pour l'exécuter.

La plupart des Shells peuvent également lire des fichiers contenant des séquences de commandes préparées. De tels fichiers sont appelé des « scripts shell ».

Un Shell utilise les étapes suivantes :

1 lire une commande du terminal ( ou d'un fichier)
2 valider la commande
3 lancer directement la commande ou démarrer le programme correspondant
4 afficher le résultat à l'écran ( ou autre part )
5 continuer à l'étape 1.

En plus de cette boucle standard, un shell contient un langage de programmation qui inclut des structures plus complexes comme les boucles, conditions et variables.

Ici, nous utiliserons le Shell Bash, l'un des plus couramment utilisé sur les systèmes GNU/Linux. Bash est un logiciel libre publié sous GNU GPL.

**Lancer un Shell (mode console ou terminal)**

Plusieurs possibilités sont offertes :

- utiliser le menu du bureau (Gnome, Kde, etc.).
- utiliser le menu lancer une application.
  - Dans la fenêtre ainsi ouverte, taper le nom de terminal et valider.
  - La fenêtre **lancer une application** peut être ouverte avec Alt+F2
- Utiliser les consoles virtuelles (il y en a 6).
  - La console virtuelle (tty1 à tty6) est un écran noir où une invite de commande apparaît, de la forme utilisateur@machine ~ $

```
    Depuis l'interface graphique, il est possible de se connecter à une
console virtuelle en utilisant
    la combinaison de touches Ctrl+Alt+FN, où N est un chiffre de 1 à 6.
    Pour revenir au mode graphique depuis une console virtuelle, utiliser la
combinaison de touches ALT+F7.
```

**EXERCICES**

- Quel est l'émulateur de terminal (aussi appelé console virtuelle ou terminal virtuel) de votre distribution ?

## 2.1.3 Les commandes.

Pourquoi des commandes?

Le fonctionnement d'un ordinateur, indépendamment de son système d'exploitation, peut être décrit en trois étapes :

1. l'ordinateur attend les données de l'utilisateur
2. l'utilisateur sélectionne une commande et l'introduit au clavier ou à la souris
3. l'ordinateur exécute la commande

Dans un système Linux, le Shell affiche un chemin (prompt), indiquant ainsi qu'une commande peut être introduite.
Ce « prompt » contient le nom de l'utilisateur, le nom de l'ordinateur sur lequel l'utilisateur est connecté, le nom du répertoire et un caractère final.

utilisateur@machine ~ $

- utilisateur → représente l'identifiant ou le nom de l'utilisateur connecté
- machine → représente le nom de la machine sur laquelle l'utilisateur est connecté
- ~ → est un raccourci qui signifie le répertoire personnel /home/utilisateur
- $ → signifie que vous êtes connecté en tant qu'utilisateur

utilisateur@machine ~ #

Si au lieu de $ le signe # apparaît, alors vous êtes connecté en tant que « superutilisateur » (root).
Les systèmes GNU/Linux utilisent par convention # pour root (utilisateur qui dispose de tous les privilèges)
et $ pour un utilisateur autre que root

**Structure d'une commande**

une commande est essentiellement un suite de mots (inspirés vaguement de l'anglais) qui se terminent par l'appui sur la touche ↩ (enter).
Pour que le Shell puisse interpréter cette suite de mots, ceux-ci doivent suivre une syntaxe.
Le premier mot de la ligne est souvent le nom de la commande et les autres mots sont les paramètres qui expliquent ce qui est demandé en détail.

A ce stade il est important de savoir que ***le shell fait la distinction entre les majuscules et les minuscules*** !
Le séparateur entre le nom de la commande et les paramètres peut être un espace ou une tabulation.
Notez que vous pouvez également utiliser ↩ (Enter) comme séparateur, à condition de le faire précéder immédiatement de \ afin que le Shell
ne l'interprète pas comme une fin de commande.

**Les paramètres d'une commande peuvent être divisées en deux types :**

- les options : elles sont précédées d'un tiret « - » et peuvent être considérées comme des « switches » qui autorisent ou pas certaines actions. Si vous souhaitez passer plusieurs options à la commande "ls", vous pouvez taper la séquence suivante «-a -s» ou «-as».
  La commande ls liste les fichiers et les sous-dossiers. (ls = liste)
- Les options longues débutent le plus souvent avec deux tirets mais ne peuvent pas être regroupées ex: « ls –all –size »

- les arguments : les paramètres non précédés de tiret sont appelés les arguments. Ils correspondent souvent au nom de fichiers que la commande doit traiter.

la structure générale d'une commande peut être expliquée comme suit :
• Command—“Que faut-il faire?”
• Options—“Comment le faire?”
• Arguments—“Sur quoi faut-il agir?”

### Types de Commandes

Dans les shells il y a essentiellement deux types de commandes :

- **Commandes Internes**. Ces commandes sont disponibles dans le Shell lui-même. Le Bash (Bourne-again shell) contient approximativement 30 commandes de ce type. Les commandes comme « exit » et « cd » modifient l'état du shell et donc ne peuvent venir de l'extérieur.
- **Commandes Externes**. Le Shell n'exécute pas directement ces commandes mais il charge des fichiers exécutables qui, dans la structure du système de fichiers Linux, sont souvent situés dans les répertoires /bin ou /usr/bin

En tant qu'utilisateur, vous pouvez utilisez vos propres programmes que le Shell exécutera en tant que commande externe.

### La complétion

Les utilisateurs d'UNIX ont de nombreux trucs pour accélérer la frappe.
Outre l'édition de ligne et son couper-coller, un truc important est la complétion: sous un shell moderne, utilisez la touche Tab pour compléter le nom d'un fichier ou d'une variable que vous avez commencé à taper. Quand il y a plusieurs fins possibles, le shell complète jusqu'à l'endroit où il faut choisir.
Dans un terminal si vous tapez “les” puis Tab, vous obtiendrez less qui permet d'afficher le contenu d'un fichier directement dans le terminal.

**Tab Tab**
Si lors du premier appui sur [Tab], le nom n'a pas été complété, un deuxième appui vous donne la liste de toues les possibilités .
**Flèche vers le haut**
La flèche vers le haut permet de remonter dans l'historique des commandes, la flèche vers le bas permet de revenir.
**Ctrl-C**
Arrête le processus en cours, celui qui a été lancé par la dernière commande.

### EXERCICES

- Ouvrez un terminal. Sur quelle machine êtes vous connecté ? Quel est le nom du répertoire courant ?
- labomons@labomons-N130:~$ echo comment ce texte va être affiché ?
- la commande « cal » est une commande interne ou externe ?
- La commande « date » affiche par exemple« dimanche 7 octobre 2012, 17:17:14 (UTC+0200) » dans le terminal. Comment faire pour, au départ de la même commande, n'afficher que le jour complet de la semaine ( lundi, mardi,…ou dimanche )?

- utilisez la commande « date » pour affichez dans le terminal le texte suivant : nous sommes un « jour complet semaine » du mois « nom complet du mois ». *Exemple: nous sommes un **dimanche** du mois **octobre**.*
- avec la commande less affichez le contenu de *source.list* qui se trouve dans le répertoire /etc/apt.(Utilisez la complétion)

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

Linux est un systeme d'exploitation puissant et complexe.
La documentation est un outil important pour gérer cette complexité, beaucoup d'aspects sont documentés de façon détaillée.
En tant qu'utilisateur Linux vous devez avoir une vue d'ensemble de la documentation disponible et connaître la façon d'obtenir de l'aide en cas d'urgence.

### 2.2.1 La Commande help , et l'option --help.

Sous bash, les commandes internes sont décrites en détails par la commande help , en donnant ce nom comme argument au nom de la commande en question.

```
$ help exit
exit: exit [n]
Termine le shell.
Termine le shell avec le code de retour N.
Si N est omis, le code de retour est celui de la dernière commande.
$ _
```

De plus amples explications sont disponibles au départ des pages du manuel shell et des sources d'informations.
Beaucoup de commandes externes (programmes) disposent de l'option **–help**.
La plupart des commandes affichent une liste brève des paramètres et de la syntaxe .
Toutes les commandes ne réagissent pas à **–help**, essayez alors **-h** or **-?**, il n'y a pas de convention universelle.

### 2.2.2 Le manuel en ligne

#### Généralités

Presque tous les programmes en ligne de commande fournissent des pages de manuel dont le texte est installé
avec le logiciel et peut être appelé avec la commande “man ⟨name⟩”.

Table 1 : Sections des pages de manuel

| Sections | Contenu |
|---|---|
| NAME | nom de commande et description brève |
| SYNOPSIS | Description de la syntaxe de la commande |
| DESCRIPTION | Description détaillée des actions de la comande |
| OPTIONS | Options disponibles |
| ARGUMENTS | Arguments disponibles |
| FILES | fichiers Auxiliaires |
| EXAMPLES | exemples de lignes de commande |
| SEE ALSO | références croisées sur des sujets en relation |
| DIAGNOSTICS | messages d'erreur et d'avertissement |

| Sections | Contenu |
|---|---|
| COPYRIGHT | Autheurs de la commande |
| BUGS | limitations connues de la commande |

"man bash", par exemple, affiche une liste des sections mentionnées ci-dessus. Les pages de manuel sont très souvent en anglais, ce qui peut constituer un désavantage.
Les sujets spéciaux sont souvent traités dans des documents "HOWTO" qui existent également en version française.

Il n'y a pas par contre d'aide pour tout ce qui appartient aux environnements graphiques KDE and GNOME.
Si cela existe les pages sont souvent mal documentées .

#### Structure

Le code source des pages de manuel est stocké dans le répertoire /usr/share/man et sous répertoire appelé man**n**, où **n** est un des chapitres de la Table 2.

On peut intégrer des pages de manuel dans des répertoires additionnels en configurant la variable d'environnement “manpath”
Cette variable contient les répertoires dans lesquels la commande “man” y recherchera dans l'ordre.

```
$ manpath
exit: exit [n]
/usr/local/man:/usr/local/share/man:/usr/share/man
$ _
```

Table 2: Sujets des pages de manuel

| Numéros | Sujet |
|---|---|
| 1 | commandes de l'utilisateur |
| 2 | appels système |
| 3 | Librairie de fonction du langage C |
| 4 | fichier des périphériques et des pilotes |
| 5 | fichiers de configuration et format des fichiers |
| 6 | jeux |
| 7 | divers (ex: tables ASCII, ...) |
| 8 | commandes de l'Administrateur |
| 9 | fonctions du noyau |
| n | ‘nouvelles’ commandes |

#### Chapitres

Les chapitres 1, 5 et 8 sont les chapitres les plus importants.
Vous pouvez donner un numéro de chapitre à la commande “man” pour affiner la recherche.
Par exemple “man **1** crontab” affiche la page du manuel de la **commande** crontab, alors que “man **5** crontab” explique le **format des fichiers** de crontab.

Lorsqu'on se réfère aux pages du manuel, il est courant d'ajouter le numéro de chapitre entre parenthèses:
crontab**(1)** affiche donc la page du manuel de la **commande** crontab, alors que “crontab**(5)**” explique le **format des fichiers** de crontab.

Avec l'option **-a** ( man -a ), la commande “man” affiche toutes les pages qui contiennent le nom indiqué.
Sans cette option, seulement les premières pages trouvées,généralement du chapitre 1 , seront affichées.

#### Afficher les pages du manuel

Le programme utilise pour afficher les pages de manuel dans un terminal est très souvent ‘less’.
A ce stade, il est important de savoir que les touches ↑ and ↓ peuvent être utilisées pour naviguerdans les pages du manuel.

La recherche d’un mot dans le manuel se fait en insérant / avant le mot à rechercher.
Une fois consulté, vous pouvez quitter le manuel d’aide avec la touche « q » pour retourner au shell.

Vous pouvez également utiliser votre navigateur favori pour consulter des pages de manuel préformatées,
il suffit d’entrer l’URL “man:/⟨name⟩” (ou “#⟨name⟩”, ou man:(name)) dans la ligne d’adresse du navigateur.

La commande “apropos” permet de chercher des informations générales sur un sujet, elle fonctionne comme man –k.
Les deux permettent d'afficher les commandes brièvement définies et en rapport avec un mot clef.
apropos linux : affiche toutes les lignes de description du champ NAME des fichiers “man” et de la base de données “whatis” qui contiennent l'expression “linux”.

la commande whatis recherche des pages de manuel dont le nom correspond à la section “NAME” et affiche leur description courte.

### Pages d'info

Pour certaines commandes, souvent plus compliquées, les pages d'info ont été développées.
Elles sont souvent plus étendues ( approfondies ) et basées sur l' hypertexte

Les pages d'info sont affichées en tapant “info (command) ”, le paquet contenant le programme associé à la commande info
peut devoir être installé de façon explicite . Les pages d'info peuvent être affichées avec votre navigateur favori, via l'url “info:/ ⟨command⟩”.

#### exercices

> afficher la page d'info de la commande “ls” dans un terminal et ensuite dans votre navigateur..
> Les “Info files” utilisent une forme brute de l' hypertext, similaire au fichiers HTML du World Wide Web.
> Pourquoi les “info files” ne sont-elles pas écrites en HTML directement ?

### HOWTOs

Tous les problèmes ne peuvent être solutionnés en utilisant l'aide associée à une commande.
C'est pour cette raison qu'il existe de la documentation orientée problèmes plutôt qu'orientée commandes.
Les HOWTOs sont développés dans ce sens. Les HOWTOs essaient d'expliquer des approches complètes pour résoudre les problèmes.

Par exemple, il existe un “DSL HOWTO” détaillant la façon de se connecter à Internet avec avec un système Linux.
Beaucoup de HOWTOs sont disponibles sous d'autres langues que l'anglais.

La plupart des distributions Linux fournissent les HOWTOs en tant que paquets à installer localement. Pour Debian, il sont situés dans le répertoire /usr/share/doc/HOWTO.
Les versions courantes de tous les HOWTOs, ainsi que d'autres formats ( PostScript ou PDF) se trouvent sur le site web
Linux Documentation Project” .

**D'autres sources d'information**

Beaucoup d'interfaces graphiques pour utilisateurs (GUI) , comme celles provenant des paquets GNOME - KDE - LXDE , proposent un menu d'aide.
Indépendemment du système local, beaucoup de documentation est disponible sur Internet :

> www.tldp.org
> www.linux.org
> doc.ubuntu-fr.org
> ftp://ftp.lip6.fr/pub/linux/french/docs/

Si vous ne trouvez pas d'information sur le Web, il est possible de poser vos questions dans des mailing listes ou dans les archives Usernet.
A ce propos avant de poser votre question, si vous ne voulez pas être mal considéré par les utilisateurs habitués, consulter la documentation
déjà disponible ou la rubrique des questions les plus souvent posées (faq); la réponse se trouve peut-être dans une rubrique.

# 2.3 Utiliser les répertoires et listings de fichiers

**Noms de fichiers**

Un des services les plus important d'un système d'exploitation comme Linux consiste à stocker des données sur des médias comme un disque dur,
une clef usb et de les récupérer plus tard. Ces données sont des collections de fichiers portant tous un nom.
Les caractères permis par Linux pour former un nom ne sont limités que par ce que votre ordinateur peut afficher.
Seuls deux caractères ne sont pas autorisés : le slash et le zero byte ( caractère ayant 0 pour valeur ASCII).

D'autres caractères comme les **espaces** ou **$** peuvent être utilisés, mais utilisés dans une ligne de commande ils doivent être “échappés” grâce au backslash
ou quotes ( guillemets ) afin d'éviter une mauvaise interprétation du shell.
Pour rappel Linux fait la distinction entre minuscules et majuscules. Donc pour Linux, les fichiers nommés “x-” et “X-” sont deux fichiers différents.
Le nom de fichier peut être long, il n'y a pas de limite supérieure; faites toutefois attention à la lisibilité du texte affiché.
Une autre différence par rapport à DOS & Windows est que Linux n'utilise pas de suffixe pour caractériser un type de fichier.

Le point est donc un caractère classique faisant partie du nom de fichier.

Toutefois, vous devez être attentif à ceci:
Le nom d'un fichier devrait toujours commencer avec une lettre ou un chiffre, les autres caractères peuvent être utilisés à l'intérieur du nom.

Exemple de noms “autorisés”

> X-files
> foo.txt.bak
> 50.zorglup
> 7_ou_10

Des problèmes pourraient survenir avec les noms de fichiers suivants :

| **noms de fichiers** | **remarques** |
|---|---|
| -10°F | **-** est un caractère spécial. |
| .profile | **.** le fichier sera caché !!!! |
| 3/4-metre | contient un caractère illégal |
| ambiguïté | Contient un tréma |

le fichier caché est souvent utilisé parce qu’il contient des paramètres pour les programmes et donc en le cachant il ne distraie pas l'utilisateur lorsqu'il affiche le contenu d'un répertoire.

**Répertoires**

Vu que plusieurs utilisateurs peuvent utiliser le même système Linux, il serait problématique que chaque nom de fichier ne pourrait être utilisé qu’une seule fois.
D’autre part l’utilisateur A doit avoir l’assurance qu’un autre utilisateur B ne puisse lire ses fichiers.

C’est la raison pour laquelle Linux a mis en œuvre une hiérarchie dans ses répertoires qui est utilisée par des groupes de fichiers.
Le nom des fichiers ne doit pas être unique dans le système complet mais uniquement à l’intérieur d’un répertoire.

Sous Linux, les répertoires sont considérés comme des fichiers, même si vous ne pouvez pas utiliser les mêmes méthodes que les fichiers pour y accéder.
Cependant les règles discutées dans la section précédente pour les fichiers s’appliquent pour les répertoires.
Le slash **« / »** est utilisé pour séparer les noms de fichier des noms de répertoire. Dans l’exemple **toto/linux.txt** signifie que le fichier **« linux.txt »**
est dans le répertoire de **« toto ».** Un répertoire peut contenir d’autres répertoires, c’est la notion de hiérarchie ou de structure arboressante.
Un système linux dispose d’un répertoire spécial qui , en quelque sorte, est la racine de l’arbre ; c’est le répertoire racine noté **« / »**(slash).

**Notion de chemin Absolu et Relatif**

Dans l’exemple **/home/toto/Linux.txt**, le nom de fichier Linux.txt est localisé dans le répertoire toto qui à son tour est localisé dans le répertoire **home**.
Ce dernier est un descendant direct du répertoire racine. Un nom de chemin qui démarre du répertoire racine est appelé nom de chemin absolu.

Chaque processus dans le système Linux a un répertoire courant ou répertoire de travail ( working directory).
Dans l'exemple **« toto/Linux.txt »**, **« Linux.txt »** est le fichier du répertoire courant **« toto »** .
De tels noms , débutant du répertoire courant, sont appelés nom de chemin relatif.

*Un nom de chemin débutant avec « / » est absolu, tous les autres sont relatifs.*

**Il y a deux raccourcis pratiques pour situer la notion de chemin.**
« .. » fait référence au répertoire situé au-dessus du répertoire en question dans l'arboressance.
Dans l'exemple **/home/toto**,**« ..»** fait référence à **/home**.

Imaginons que /home/toto dispose de deux sous-répertoires nommés « scripts » et « programmes »
Si vous êtes dans le répertoire courant « scripts », pour faire référence au fichier python.txt, vous pouvez utiliser « ../programmes/python.txt »
sans avoir à utiliser le chemin absolu /home/toto/programmes/python.txt

Le second raccourci est «.» et, dans un répertoire, il signifie le répertoire lui-même.
Quel est son intérêt ?
Lorsque le Shell recherche des programmes associés à des commandes externes, il le fait dans les répertoires indiqués dans la variable PATH.
Si vous voulez invoquer un programme « prog » inclus dans un fichier lié au répertoire courant mais que ce répertoire n'est pas référencé dans le
PATH, vous pouvez demander au Shell de lancer votre fichier en tant que programme en écrivant après le prompt « ./prog » sans avoir à introduire le
chemin absolu !

### Commandes pour les répertoires.

#### Le répertoire courant : cd & Co.

Vous pouvez utiliser « cd » (Change Directory) pour changer de répertoire courant .

$ cd toto le répertoire courant devient "toto"
$ cd .. le répertoire courant devient celui situé avant celui dans lequel vous vous trouviez.

Si vous ne fournissez pas de paramètre à la commande « cd » , vous resterez dans votre répertoire de travail « home ».
$ cd
$ « cd »|pwd]]
/home/toto

Vous pouvez afficher le chemin absolu du répertoire courant en utilisant la commande « cd »|pwd]] (print working directory)
Si dans le « prompt » vous voyez le caractère « ~ », cela signifie que vous êtes dans le répertoire courant.
La commande « cd – » permet de basculer vers la dernière commande « cd » que vous avez effectuée.
Cela peut être intéressant à utiliser pour alterner entre deux répertoires.

### Exercices.

> La commande « cd » est-elle une commande externe ou interne du shell ? Pourquoi ?

> Lisez les pages du manuel des commandes « push », « popd » et « dirs ». Quelles sont leurs utilités respectives ?

### Lister les fichiers et répertoires – ls

La commande « ls » (list) permet de trouver son chemin dans l'arboressance des répertoires.
Utilisée sans option, elle affiche le nom des fichiers sur plusieurs colonnes.
La plupart des distributions se sont mise d'accord sur la couleur à utiliser pour représenter les fichiers, les répertoires....

Table 1: quelques exemples de types de fichiers pour la commande « ls »

| types de fichiers | Couleur | Suffixe (ls -F) | Type de lettre (ls -l) |
|---|---|---|---|
| plain file | noire | rien | - |
| fichiers exécutables | verte | * | - |
| répertoires | bleue | / | d |
| liens | orange | @ | l |

Table 2: quelques options de la commande « ls »

| Options | Résultats |
|---|---|
| -a ou –all | affiche également les fichiers cachés |
| -i ou –inode | affiche le numéro de fichier unique (inode number) |
| -l ou –format=long | affiche des informations supplémentaires |
| -o ou –no-color | enlève le codage de couleur des résultats |
| -p ou -F | marque les types de fichier en ajoutant un caractère spécial |
| -r ou –reverse | effectue un tri inversé |
| -R ou –recursive | tri récursif dans les sous répertoires (DOS: DIR/S) |
| -S ou –sort=size | les fichiers sont triés par taille |
| -t ou –sort=time | les fichiers sont triés en fonction de la date et de l'heure |
| -X ou –sort=extension | les fichiers sont triés par type |

**Exemple de la commande ls –l (format =long )**
-rw-r–r– 1 toto users 4711 Oct 4 11:11 file.txt
-rw-r–r– 1 toto users 333 Oct 2 13:21 file2.dat

Signification des différentes parties :

> Le premier caractère :
>> «-» = fichier
>> « d » = répertoire
>
> Les 9 caractères suivants indiquent les permissions d'accès aux fichiers
> Les caractères suivants indiquent le propriétaire du fichier « toto » et le groupe auquel appartient le fichier « users »
> Après la taille exprimée en Byte, sont affiché la date et l'heure de la dernière modification faite dans le fichier.
> Le nom de fichier est indiqué à la fin.

**Exercices :**

> - Quels fichiers contient le répertoire « /boot » ? Y-a-t-il des sous répertoires et si oui, lesquels ?
> - Expliquez la différence entre la commande « ls » avec pour argument un nom de fichier ou un nom de répertoire ?
> - Comment avec la commande « ls » afficher les informations à propos d'un répertoire au lieur des informations des fichiers de ce répertoire, si vous passez le nom du répertoire au programme ?

# 2.4 Créer, déplacer et effacer des répertoires ou des fichiers

La commande **« mkdir »** permet de créer de nouveaux répertoires;elle nécessite d'ajouter,comme argument,le nom d'un ou plus noms de répertoire.

**Pour créer des répertoires** imbriqués en une fois, vous utiliserez le paramètre **« -p »**.

```
$ mkdir toto/photos
mkdir: cannot create directory `pictures/holiday': No such file_
_ or directory
$ mkdir -p toto/Photos
$ cd toto
$ ls –F
Photos/
```

**Pour supprimer un répertoire**, utiliser la commande **« rmdir »**(remode directory)

Tout comme pour la commande « mkdir » elle nécessite d'ajouter, comme argument, le nom d'un ou plus nom de répertoire à supprimer.
Mais le répertoire doit être vide (pas de fichier, sous-répertoires..) !

```
$ rmdir -p toto/Photos
$ ls –F
toto/
```

Avec l'option –p, tous les répertoires vides mentionnés dans le nom de l'argument seront effacés en une seule étape, en commençant par le nom le plus à droite.

```
$ mkdir -p toto/Photo/vacances
$ rmdir toto/Photo/vacances
$ ls -F toto
Toto/Photo/
$ rmdir - toto/Photo
$ ls -F toto
ls: pictures: No such file or directory
```

**exercices**

Dans le répertoire « home », créer un sous-répertoire « test » qui contiendra les répertoires « rep1 », « rep2 », « rep3 ».

Placez-vous dans le répertoire « rep1 » et créer (avec par exemple un éditeur de texte) un fichier texte nommé « FichierTexteRep1 »
et contenant le texte « FichierTexteRep1».
Dans le répertoire « rep2 », créer un fichier « FichierTexteRep2 » contenant le texte « FichierTexteRep2 ».

> Vérifier que les fichiers créés existent !
> Supprimer le répertoire « rep3 »
> Supprimer le réperoire « rep2 » ; Que se passe-t-il, et Pourquoi ?

### Caractères de remplacement pour la recherche de fichiers

#### caractères simple de remplacement

« * » le shell remplace l'astérisque par une liste triée de nom de fichiers qui est associé à ce paramètre. L'astérisque remplace un ou plusieurs caractères Une commande Shell comme « prog/p*.c » retournera par exemple : prog/p1.c prog/polly.c prog/pop-rock.c prog/p.c

Seul le caractère "/" ne sera pas filtré par « * » ; il est donc préférable de limiter la recherche avec « * » dans le répertoire courant. Par défaut, la recherche par « * » n'affiche pas les fichiers cachés commençant par « . »
Pour les inclure on pourrait utiliser « .* »

« ? » remplace un seul caractère qui doit exister

Les caractères de remplacement sont sous la responsabilité du Shell. Dès lors si aucun fichier n'est associé à la recherche, le shell renverra un message d'erreur !

#### classes de caractères

Un filtre de recherche de la forme « Prog[123].c » retournera uniquement les fichiers Prog1.c Prog2.c Prog3.c

Un filtre de recherche de la forme « Prog[1-9].c retournera les fichiers Prog1.c jusque Prog9.c

Un filtre de recherche de la forme « Prog[A-z].c retournera les fichiers dont le caractère est compris entre A et z de la table ASCII.
Dès lors le fichier « Prog@.c » sera filtré !

Un filtre de recherche de la forme « Prog[A-Za-z].c retournera les fichiers dont la dernière lettre est une majuscule ou une minuscule.

Un filtre de recherche de la forme « Prog[ !A-Za-z].c retournera les fichiers dont le dernier caractère n'est pas une lettre.
Comme d'habitude le slash « / » est une exception.

# 3 La puissance de la ligne de commande

# (pondération : 10)

## 3.1 Sauvegarder des fichiers en ligne de commande

## 3.2 Recherche et extraction de données de fichiers

## 3.3 Convertir des commandes en script

# 4 Le système d'exploitation Linux (pondération : 8)

## 4.1 Choisir un système d'exploitation

## 4.2 Comprendre le fonctionnement matériel d'un ordinateur

## 4.3 Les endroits où sont stockées les données

## 4.4 Votre ordinateur sur le réseau

# 5 Sécurité et permissions de fichiers (pondération : 7)

## 5.1 Bases de la sécurité et identification des types d'utilisateurs

## 5.2 Créer des utilisateurs et des groupes

## 5.3 Gérer les permissions et propriétaires des fichiers

## 5.4 Répertoires et fichiers spéciaux

# Références

- http://www.lpi.org/linux-certifications/introductory-programs/linux-essentials
- Annonce
- Document de préparation (version anglaise en bas de page), non libre !
- The Linux System Administrator's Guide, document sous “GNU Free Documentation License”
- http://www.linuxquestions.org
- Linux par l'exemple : Notions de base et configurations types, document sous licence GPL de 1999

From:
https://www.loligrub.be/wiki/ - **LoLiGrUB**

Permanent link:
**https://www.loligrub.be/wiki/lpic:lpi_linux_essentials?rev=1351450476**

Last update: **2014/12/27 08:14**