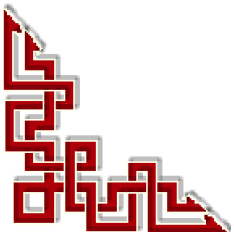
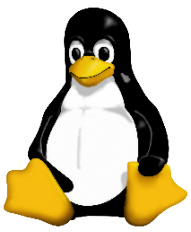
RF-232

Micronator-dev

Linux-101

Cahier-01





Version: 3.0.1 / jeudi, 9 janvier 2025 - 14:42

AVIS DE NON-RESPONSABILITÉ

Ce document est uniquement destiné à informer. Les informations, ainsi que les contenus et fonctionnalités de ce document sont fournis sans engagement et peuvent être modifiés à tout moment. ***RF-232*** n'offre aucune garantie quant à l'actualité, la conformité, l'exhaustivité, la qualité et la durabilité des informations, contenus et fonctionnalités de ce document. L'accès et l'utilisation de ce document se font sous la seule responsabilité du lecteur ou de l'utilisateur.

RF-232 ne peut être tenu pour responsable de dommages de quelque nature que ce soit, y compris des dommages directs ou indirects, ainsi que des dommages consécutifs résultant de l'accès ou de l'utilisation de ce document ou de son contenu.

Chaque internaute doit prendre toutes les mesures appropriées *(mettre à jour régulièrement son logiciel antivirus, ne pas ouvrir des documents suspects de source douteuse ou non connue)* de façon à protéger le contenu de son ordinateur de la contamination d'éventuels virus circulant sur la Toile.

Toute reproduction interdite

Vous reconnaissez et acceptez que tout le contenu de ce document, incluant mais sans s'y limiter, le texte et les images, sont protégés par le droit d'auteur, les marques de commerce, les marques de service, les brevets, les secrets industriels et les autres droits de propriété intellectuelle. Sauf autorisation expresse de ***RF-232***, vous acceptez de ne pas vendre, délivrer une licence, louer, modifier, distribuer, copier, reproduire, transmettre, afficher publiquement, exécuter en public, publier, adapter, éditer ou créer d'oeuvres dérivées de ce document et de son contenu.

Avertissement

Bien que nous utilisions ici un vocabulaire issu des techniques informatiques, nous ne prétendons nullement à la précision technique de tous nos propos dans ce domaine.

En un clin-d'oeil

Sommaire

I- Description générale

1. Introduction

Pour connaître les rudiments de Linux ou rafraîchir vos connaissances, vous pouvez consulter ce document: *Cours Linux-101*. Ce cours donne un aperçu des fonctionnalités de base de Linux qui sont indispensables à toute personne qui désire se familiariser à l'environnement Linux et surtout comprendre et maîtriser les concepts de base.

Vous serez en mesure de recourir à la documentation en ligne (*man*), manipuler l'arborescence des fichiers, comprendre l'organisation générale du système, gérer les droits d'accès, découvrir les variables d'environnement, les fichiers particuliers, la quincaillerie réseau, utiliser les principales commandes bash, etc.

Un chapitre particulier, qu'il n'est pas nécessaire de maîtriser mais simplement connaître, explique les principes de base de la communication **TCP/IP**.

Enfin, une introduction à l'éditeur **vi** est donnée à la fin du document.

Référence: Ce cahier est une adaptation de la page **http://www.linux-france.org/article/kafkafr/**.

Autre référence: **free-electrons.com/doc/legacy/command-line/unix_linux_introduction_fr.pdf**.

2. Conventions de ce document

2.1. Notes au lecteur

* Les captures d'écrans ne sont que des références.

** Les informations écrites ont préséance sur celles retrouvées dans les captures d'écrans. Se référer aux différents tableaux lorsque ceux-ci sont présents.

2.2. Conventions

Toutes les commandes à entrer à la console sont en **gras**. Les affichages à surveiller sont en **rouge**, **bleu**, **orange**, **vert** ou **magenta**.

```
# ping 192.168.71.62

192.168.71.62 is alive
#
```

Les liens de référence Internet sont en **bleu** et ceux intradocument en ***bleu***.


Manipulation, truc ou ruse pour se tirer d'embarras.


Une recommandation ou astuce.


Une note.


Une étape, note ou procédure à surveiller.


Paragraphe non complété ou non vérifié.


Danger pour la sécurité du système.


Dans un commande, indique qu'il faut remplacer le texte en **magenta** par votre propre texte.


Cette icône indique que la commande est sur une seule ligne. Ce document en PDF l'étalera sur deux lignes avec un [CR] [LF] entre elles. Il faudra donc copier la commande entière dans un éditeur de texte ASCII et la mettre sur une seule ligne avant de la copier/coller à la console.

Une **chaîne de caractères en magenta** indique qu'il faut remplacer cette chaîne par vos propres paramètres.

```
Commande à exécuter si ce n'est déjà fait.
```

```
Commande indiquée à titre d'information seulement.
```


II- Linux

1. Aperçu général

Nous présentons ici ce qu'est Linux, d'où il vient et comment il est distribué actuellement. Pour ceux ayant déjà une connaissance d'autres systèmes d'exploitation, nous proposons également une comparaison avec quelques-uns d'entre eux.

2. Un peu d'histoire... UNIX!

2.1. Caractéristiques principales

Linux est un système multitâche. Cela signifie que plusieurs programmes peuvent s'exécuter simultanément; par exemple, vous pourriez rédiger une lettre sur un traitement de textes, tout en attendant qu'un programme d'images de synthèse termine son calcul, et ce en même temps qu'un **CD** diffuse une musique plus ou moins mélodieuse. Il n'y a pas de limite théorique au nombre d'applications qui peuvent ainsi fonctionner simultanément: les facteurs limitants sont en premier lieu la quantité de mémoire dont dispose l'ordinateur, et en second lieu, la puissance du processeur. Pour fixer les idées, il est très courant d'avoir une cinquantaine de programmes (*on dit aussi "processus"*) s'exécutant sous Linux dès qu'une couche graphique est démarrée.

Ensuite, Linux est multi-utilisateur. C'est-à-dire que tout est prévu pour que plusieurs personnes utilisent le système, éventuellement simultanément (*si le "câblage" le permet*). En pratique, chaque personne utilisant le système dispose d'un compte, qui peut être vu comme une certaine zone qui lui est allouée, accessible par un nom et un mot de passe. Elle ne peut normalement pas en sortir, et encore moins aller modifier (*volontairement ou non*) les zones allouées à d'autres personnes. Il existe un mécanisme de droits permettant d'assurer ce contrôle permanent qui peut paraître contraignant mais évite bien des erreurs.

Linux est de plus fortement "orienté réseaux": il offre tous les outils et tous les mécanismes pour la mise en place de réseaux locaux, l'intégration dans des réseaux existants, la connexion à l'Internet, que ce soit comme source, traiteur ou collectionneur d'informations. Héritée de son inspirateur Unix, cette caractéristique fait de Linux l'un des systèmes les plus ouverts qui soit.

Enfin, et c'est probablement ce qui fait sa plus grande force, Linux est **LIBRE** (*Open Source*): chacun peut examiner le code en langage **C** du système, le modifier selon ses besoins, proposer des ajouts ou des améliorations... Aujourd'hui, plusieurs milliers de programmeurs, communicant par l'Internet, participent au développement de Linux. Dès qu'une nouvelle version apparaît, des milliers d'utilisateurs l'installent, la testent, signalent les erreurs. Une entreprise ne peut qu'approcher de la force colossale que représente cette masse.

Ajoutons que s'il fut initialement développé pour les ordinateurs de types **PC**, Linux existe également sur de nombreuses autres plateformes, à base de processeurs **PowerPC**, **Alpha**, **68K**, et d'autres. À l'heure où nous écrivons ces lignes, il existe même un projet de portage pour le **PalmPilot** de **3Com**!

2.2. Les distributions

Si l'on veut être rigoureux, le terme "Linux" en lui-même ne désigne que le coeur du système (*appelé le noyau, ou plus communément le kernel*), cette zone un peu obscure qui permet d'utiliser le matériel et de communiquer avec le reste du monde. Mais par lui-même, Linux serait bien incapable d'enregistrer le moindre texte tapé au clavier!

La méthode usuelle pour obtenir Linux est d'acquérir l'une des nombreuses distributions qui existent. Une distribution est un assemblage, autour d'un noyau Linux, d'un ensemble plus ou moins vaste de programmes utilitaires permettant d'utiliser toute la puissance de Linux. En général, ces distributions sont gratuites, et se présentent soit sous forme de **CD-ROM**, soit de disquettes (*pour les petites...*), et sont même parfois directement téléchargeables depuis l'Internet (*ce qui peut prendre beaucoup de temps!*)

Il est possible d'obtenir une distribution, soit en la commandant directement auprès du fournisseur (*en général par l'Internet*), soit en l'achetant en magasin spécialisé. Ce que l'on paye alors correspond le plus souvent à l'emballage et à la documentation imprimée qui accompagnent le **CD-ROM**... Toutefois, il existe des distributions payantes, qui ne sont alors pas librement re-distribuables. Enfin, il n'est pas rare de trouver une distribution pour accompagner un ouvrage traitant de Linux, ou une simple revue disponible en bureau de presse...

Citons quelques-unes des distributions Linux les plus célèbres: **RedHat**, **CentOS**, **NethServer**, **SuSE**, etc.

3. Système de fichiers

Le système de fichiers d'un système d'exploitation est un ensemble de principes et de règles selon lesquels les fichiers sont organisés et manipulés. Chaque système d'exploitation possède son système de fichier privilégié, même s'il peut en utiliser d'autres. Le tableau ci-dessous donne quelques noms.

Système	Système de fichier
MS-DOS	FAT, aussi appelé FAT-16 (File Allocation Table 16bits)
Windows 95/98	FAT32, extension du système FAT-16 (File Allocation Table 32bits)
Windows NT/7/8.x	NTFS (New Technology File System)
OS/2	HPFS (High Performance File System)
Linux	Ext2fs (Second Extended File System), Ext3, Ext4

Tableau 1: Systèmes de fichiers.

3.1. Qu'est-ce qu'un fichier?

Fondamentalement, un fichier est une suite de caractères (*on dit "octets"*) qui constitue un ensemble cohérent (*en principe*) d'informations. Le fichier est l'élément fondamental d'un système d'exploitation: les instructions nécessaires au fonctionnement de ce système, sa configuration, comment il doit réagir à telle situation, toutes ces informations sont stockées dans des fichiers.

Plus généralement, les fichiers sont utilisés pour stocker les travaux des utilisateurs. Pour ces derniers, les fichiers sont identifiables par un nom, qui doit respecter certaines contraintes (*voir plus bas*). Cette définition, très générale, inclut également les fichiers qui rassemblent des informations sous forme structurée selon les principes des bases de données, par exemple le fichier des assurés sociaux de la Sécurité Sociale: c'est là une forme plus restrictive de la notion de fichier.

Enfin, les fichiers sont eux-mêmes stockés physiquement sur différents supports, comme les disquettes, les disques durs, les **CD-ROMs**...

3.2. Tout est fichier!

Il en est ainsi sous Unix, et donc sous Linux: tous les éléments du système sont manipulés par des fichiers! Cela peut paraître un peu déroutant de penser que le clavier sur lequel on frappe désespérément est considéré par le système comme un fichier, spécial certes, mais fichier tout de même. Toutefois cette uniformisation permet de simplifier grandement la manipulation du système en général, et sa programmation en particulier.

Sous Linux, les noms de fichiers sont limités à **255 caractères** et ne doivent en principe comporter que des caractères alphanumériques (*c'est-à-dire des lettres et des chiffres*), ainsi que les symboles **. - _ ~ + %**. Il est toutefois possible d'insérer un espace ou d'autres caractères "bizarres" dans un nom de fichier, mais ceci est très fortement déconseillé.

Attention, le système fait la différence entre les majuscules et les minuscules! Ainsi, dans un même répertoire on peut avoir les deux fichiers **lettre.txt** et **Lettre.txt**. Nous ne saurions trop vous déconseiller d'abuser de ce genre de possibilités.

Une petite particularité: les fichiers (*ou répertoires*) dont le nom commence par un point ('**.**') sont des fichiers cachés, c'est-à-dire qu'ils ne sont pas listés normalement par la commande **ls**.

Vous aurez sans doute remarqué que nous utilisons une forme de caractère particulière pour les noms de fichiers, ceci afin de faciliter la lecture: il en sera ainsi tout au long de ce document.

4. Les répertoires

Les répertoires sont des fichiers un peu particuliers, qui "contiennent" d'autres fichiers et d'autres répertoires.

Le terme de "contenir" n'est pas tout à fait approprié, ce n'est qu'une image. Plus précisément, un répertoire est un fichier contenant des références et des descriptions pour d'autres fichiers (*qui peuvent être des répertoires...*). Ce terme est toutefois celui employé couramment.

On obtient ainsi une structure arborescente, telle que celle visible sur la ***Figure 1***. L'utilisation de répertoires est vivement conseillée pour organiser ses informations.

Il existe sous Linux (*et Unix*) un répertoire "ancêtre" de tous les autres, qui contient tous les autres (*directement ou suite à des intermédiaires*): le répertoire **root**, que l'on note simplement '/' (*ce symbole se lit barre oblique ou slash*).

Prenons une analogie, pour bien faire saisir cette notion de répertoire. Considérez votre système comme une bibliothèque: la bibliothèque dans son ensemble peut être assimilée au répertoire racine. Mais généralement, une bibliothèque est subdivisée en sections: la section "histoire", la section "géographie", la section "roman"... ce sont là les répertoires situés dans la racine. Entrons dans la section "roman". On y trouve d'autres divisions, les romans "policiers", les romans "picaresques", les romans "à l'eau de rose"... ce sont des sous-répertoires du répertoire "roman". Mais peut-être y a-t-il des romans inclassables, qui n'entrent dans aucune des divisions prévues: ils seront simplement entassés à l'entrée de la section "roman", directement accessibles, comme les fichiers présents dans un répertoire. Au sein des romans policiers, on peut trouver encore d'autres catégories selon la langue, l'auteur, etc. pour parvenir enfin jusqu'à une étagère ou un rayon, qui ne soit pas subdivisé: les rayons sont remplis de livres, qui sont les fichiers, car comme les fichiers ils ont un nom (*le titre*) et un propriétaire (*l'auteur*).

Chaque répertoire contient deux répertoires particuliers:

'**.**'

désigne le répertoire lui-même et

'**..**'

désigne le répertoire de niveau juste supérieur.

Ainsi, sur la ***Figure 1***, le '**..**' du répertoire **rc.d** désigne le répertoire **etc**.

Une notion importante sous Linux est celle de répertoire courant. On peut considérer une arborescence comme un ensemble de villes reliées par des routes; le répertoire courant désigne alors la ville où l'on se trouve. Pour reprendre l'analogie de la bibliothèque, le répertoire courant est la section ou sous-section dans laquelle on est en train de déambuler nonchalamment.

Pour définir complètement un fichier, il est nécessaire de connaître la succession de répertoires qu'il faut parcourir pour le trouver: c'est le **chemin** du fichier.

Si ce chemin commence par le symbole '/', il est dit absolu: il indique la succession de répertoires depuis la racine pour obtenir le fichier. Par exemple, le texte "**/etc/passwd**" est le chemin pour le fichier **passwd** qui se trouve dans le répertoire **etc**. Par abus de langage, on appelle également chemin absolu le texte "**/usr/local/man**", qui ne désigne pas un fichier particulier, car comme on le voit sur la ***Figure 1***, **man** est un répertoire dans **/usr/local**. Mais rappelons que les répertoires ne sont que des fichiers particuliers...

Dans le cas contraire, le chemin est dit **relatif**: il prend comme point de départ le répertoire courant. Ainsi, si le répertoire courant est **/usr/local**, le texte **man/man1** équivaut au chemin absolu **/usr/local/man/man1**. En utilisant le '**..**' évoqué plus haut, en partant du même point de départ, le texte **../bin** équivaut au chemin absolu **/usr/bin**. On parle également d'indirection, lorsque le chemin commence par un '**..**'.

Ces principes et notions peuvent paraître déroutantes dans un premier temps, mais un peu de pratique les font devenir une seconde nature.

5. Hiérarchie standard

Il existe une certaine normalisation dans l'architecture des répertoires sous Linux (*comme sous Unix, d'ailleurs*). La ***Figure 1*** présente une organisation standard. Il peut toutefois exister des variations d'une distribution à l'autre, mais dans l'ensemble c'est assez bien respecté.

Les flèches en pointillés indiquent qu'il est fréquent de trouver d'autres répertoires, mais sans qu'il existe véritablement de norme.

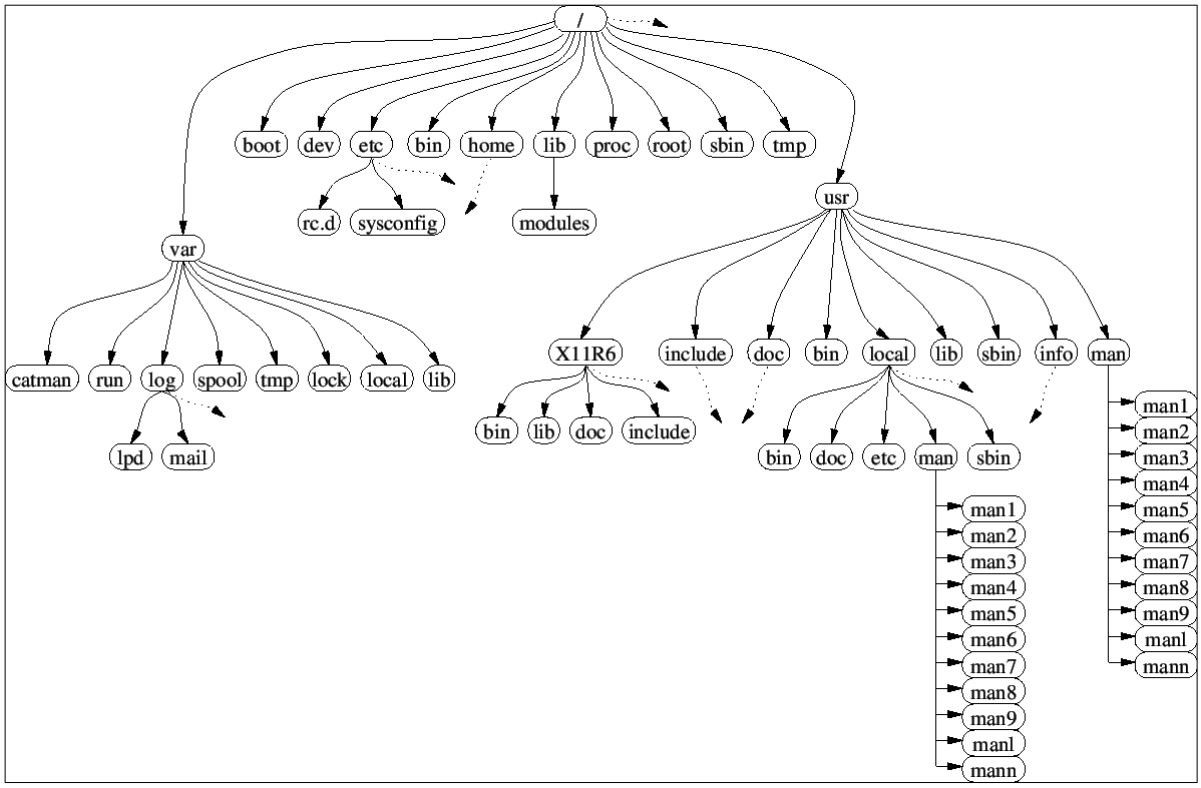


Figure 1: Hiérarchie standard des répertoires.

5.1.1. Quelques précisions concernant la Figure 1

- **/etc**

  Contient les fichiers généraux de configuration, les commandes à exécuter au démarrage du système et même le mode de démarrage du système.

- **/bin**

  Contient des commandes nécessaires lors du démarrage du système. Ces commandes pourront par la suite être utilisées par les utilisateurs.

- **/sbin**

  Contient des commandes nécessaires lors du démarrage du système, mais en quelque sorte réservées à l'administrateur du système ou super-utilisateur.

- **/home**

  Est réservé à l'hébergement des comptes des utilisateurs.

- **/dev**

  Ce répertoire contient tous les fichiers spéciaux utilisés pour accéder au matériel qu'il s'agisse: du clavier, du disque dur, de la carte son...

- **/proc**

  Voici un pseudo-répertoire: en réalité, son contenu n'existe pas physiquement sur le disque, mais est manipulé directement par le système lui-même. On y trouve nombre d'informations techniques utiles.

- **/usr**

  Répertoire à usages multiples, dont les principaux sont:

  /usr/bin
  Commandes utilisables par tous les utilisateurs, et non nécessaires lors du démarrage du système.

  /usr/sbin
  Commandes réservées au super-utilisateur, et non nécessaires lors du démarrage du système.

  /usr/man
  Contient les pages de manuel.

  /usr/doc
  Contient de nombreuses documentations et sources d'informations.

  /usr/X11R6
  Contient (*normalement*) tous les fichiers se rapportant à la couche graphique **X-Window**.

  /usr/local
  Lieu où sont stockés les fichiers spécifiques au système installé.

- **/var**

  Contient des données mises à jour par différents programmes durant le fonctionnement du système.

  /var/lock
  Fichiers de blocage, pour interdire par exemple deux utilisations simultanées du modem.

  /var/spool
  Répertoires utilisés pour l'organisation du travail des imprimantes, de la messagerie électronique, etc.

  /var/log
  Contient les fichiers journaux (*logs*) provenant de différents points du système.

6. Quelques fichiers sensibles

Ces fichiers sensibles sont des fichiers particulièrement importants, sur lesquels repose une grande partie de la stabilité du système, voire de son simple fonctionnement. Il est donc vivement recommandé de ne pas modifier, encore moins d'effacer, ces fichiers, sans un luxe de précautions et sans savoir très précisément ce que l'on fait.

- **/etc/passwd**

  Le fichier des utilisateurs et des mots de passe associés. Supprimez ce fichier, et il vous sera impossible d'utiliser votre système!

- **/etc/inittab**

  Précise le mode de démarrage du système et les actions associées. La suppression ou une altération malheureuse de ce fichier peut rendre le système impossible à démarrer.

- **/etc/fstab**

  Liste des partitions utilisées par le système et selon quelle méthode il les utilise. À ne pas toucher sans savoir! Vous pourriez rendre votre système très instable.

- **XF86Config**

  Peut se situer en différents endroits (*mais souvent sous **/etc/X11***) selon la distribution employée. Contient la configuration de la couche graphique. Attention, les informations qu'il contient sont spécifiques au matériel utilisé: une modification hasardeuse peut rendre la couche graphique non fonctionnelle, voire endommager le matériel, notamment l'écran.

- **vmlinuz...**

  Peut-être le plus important de tous. Se situe généralement, soit sous la racine (/), soit sous **/boot**. En fait, il s'agit du système lui-même! C'est le coeur des fonctions internes qui réalisent le fonctionnement de tout le système.

Ne sont évoqués ici que les fichiers les plus importants. Il existe naturellement un grand nombre d'autres fichiers, qu'il convient de traiter avec douceur, même si les effets d'une erreur sont moins catastrophiques.

7. Accéder au système et le quitter proprement

7.1. Procédure d'accès au système

Sauf circonstance exceptionnelle, l'accès à un système Linux se fait par une procédure d'identification, en fournissant un mot de passe: Linux est en effet protégé, dans le sens où seules les personnes autorisées peuvent l'utiliser. Celles-ci sont appelées les utilisateurs du système. L'ensemble de cette procédure (*nom et mot de passe*) est appelée communément **procédure de login**, ou simplement **login**. Bien que cela ne soit pas très français, on dit aussi d'un utilisateur qu'il se "logue" sur le système.

Pour l'instant, bornons-nous à décrire brièvement la connexion au système. Lorsque vous démarrez Linux, après un défilement plus ou moins long et ésotérique d'informations diverses, une ligne avec un curseur attend une entrée de votre part, par exemple:

```
login as:
```

Le terme **login** fait référence à l'identification attendue: à cet endroit, vous devez taper un nom d'utilisateur, validé par la touche [**Entrée**]. Pour un système qui vient d'être installé, ce nom est généralement **root**.

```
login as: root
```

Puis apparaît la ligne:

```
root@192.168.1.156's password:
```

Le premier mot est le nom utilisé pour se loguer suivi du caractère "**@**" et le **nom du domaine** du serveur ou son **adresse IP**. Enfin le mot **password:**.

A cet endroit vous donnez le mot de passe correspondant à l'utilisateur indiqué précédemment. Il est normal de ne rien voir s'afficher à l'écran: c'est pour éviter que quelqu'un ne lise votre mot de passe par-dessus votre épaule. Validez par [**Entrée**], et vous pouvez alors commencer à utiliser le système.

8. Notion de console

Considérez un ordinateur sur lequel tourne, par exemple, Linux. Ce que l'on appelle la console, c'est l'ensemble écran-clavier-souris, directement branchés sur l'ordinateur en question. La console se distingue du terminal (*constitué des mêmes éléments*) en ceci que ce dernier accède à l'ordinateur par l'intermédiaire d'un réseau.

Physiquement, il ne peut y avoir qu'une seule console pour un ordinateur donné (*sauf de très rares exceptions*). Linux, toutefois, offre la possibilité de "faire comme si" l'ordinateur disposait de plusieurs consoles: ces consoles sont alors dites virtuelles, car à un instant donné, seule l'une d'entre elles correspond à une réalité physique. Dans la littérature que vous pouvez rencontrer, il n'est pas rare de voir désigner les consoles par le terme "**TTY**" (*qui vient du mot "**teletype**"*).

En mode texte, ces consoles sont habituellement au nombre de **6**. Ce nombre peut varier, être adapté à vos besoins spécifiques, selon le contenu du fichier **/etc/inittab**.

Le passage d'une console virtuelle à l'autre peut se faire à n'importe quel moment durant le fonctionnement du système: il suffit, au clavier, de presser simultanément la touche [**Alt**] et l'une des touches de fonction [**Fn**], où **n** est le numéro de la console virtuelle que vous souhaitez obtenir. Au démarrage, vous êtes sous la console **1**.

Ce mécanisme peut vous paraître un gadget, mais son utilité vous apparaîtra rapidement à l'usage. Une utilisation typique est d'utiliser une console pour saisir et exécuter des commandes plus ou moins complexes, et une ou plusieurs autres consoles pour visualiser diverses documentations expliquant comment utiliser ces commandes.

Nous n'avons évoqué ici que le cas du mode texte. Un mécanisme analogue existe pour le mode graphique, plus sophistiqué (*sans pour autant être plus complexe*).

9. Quitter le système

Au stade actuel de son développement, Linux est suffisamment robuste pour fonctionner vingt-quatre heures sur vingt-quatre, pour une durée théoriquement infinie (*sauf dysfonctionnement grave d'un programme, au point de déstabiliser le système*). Mais pour une machine personnelle, il est plus courant de ne l'allumer que lorsqu'on en a besoin, et de l'éteindre aussitôt que l'on a terminé ce que l'on avait à faire.

Du fait de sa complexité, Linux ne doit pas être arrêté brutalement, sous peine de risquer d'endommager gravement le système dans son ensemble (*ceci est également vrai pour d'autres systèmes, tels que Windows ou Unix*): il est nécessaire, en quelque sorte, de prévenir Linux que l'on s'apprête à stopper son fonctionnement. Ceci afin de lui permettre de s'assurer que tout ce qui devait être sauvegardé l'a bien été, et plus généralement de prévenir les éventuels utilisateurs ou machines qui lui sont connectés.

L'arrêt immédiat du système se fait par l'une des deux commandes, équivalentes:

```
shutdown -h now
```

```
halt
```

Si l'on désire non seulement arrêter immédiatement le système, mais en plus redémarrer la machine juste après cet arrêt, l'une des deux commandes suivantes peut être utilisée:

```
shutdown -r now
```

```
reboot
```

Seul l'utilisateur **root** est normalement autorisé à utiliser ces commandes, mais certains environnements graphiques permettent à d'autres utilisateurs d'arrêter la machine.

Nous n'avons évoqué ici que les procédures d'arrêt les plus simples. Vous pouvez toutefois avoir besoin d'un comportement plus sophistiqué, selon vos besoins: consultez les pages **man** de **shutdown** pour plus de détails.

10. Le shell

Le **shell** est l'interpréteur de commandes. C'est un programme lancé juste après la procédure de **login** (*voir plus haut*), dont le rôle est d'interpréter et d'exécuter les diverses commandes que vous pouvez lui donner: c'est en quelque sorte lui qui va réaliser la communication entre vous et le système.

En fait, "**shell**" est un terme générique: il existe en effet plusieurs types de shell, chacun ayant sa propre syntaxe, son propre langage de programmation - car on peut écrire de véritables programmes pour le shell, pour des opérations trop sophistiquées pour la ligne de commande.

Le shell standard (*aussi appelé **shell** de base*) fait partie des éléments fondamentaux du système et doit donc toujours être présent. C'est une reproduction fidèle du shell standard qui accompagne tout système Unix: il est assez fortement normalisé, ce qui permet d'écrire des programmes relativement portables d'un système à l'autre. On le désigne simplement par "**sh**", du nom du programme correspondant.

Sous Linux, on utilise plutôt une extension de ce shell, le **shell bash** (*pour **"Bourne Again SHell"**, du nom du créateur du shell standard qui développa également cette extension*). En fait, **bash** est considéré comme le shell standard sous Linux: il respecte complètement la norme standard, avec quelques fonctionnalités en plus.

De nombreux autres shells existent. Citons les plus répandus, le **C-shell** et le **KornShell**.

Le **C-shell** (*ou **csh***), comme son nom l'indique, présente une syntaxe proche de celle du langage **C**. Sans doute plus complexe à utiliser que le shell standard, les possibilités offertes sont toutefois plus vastes. C'est le deuxième shell le plus utilisé sous Linux, derrière **bash**.

Le **KornShell** (*ou **ksh***) peut être vu comme une mixture du **C-Shell** et du **Bourne Shell**. Sa richesse n'a d'égale que sa complexité, c'est pourquoi il est assez peu utilisé. Mais il est bon de le connaître, même vaguement, ne serait-ce que pour votre culture personnelle.

En pratique, seule une connaissance générale du **Bourne Shell** est indispensable: c'est en effet lui qui est utilisé pour exécuter la très grande majorité des scripts de configurations, si ce n'est tous. Mais gardez à l'esprit que nombre d'outils graphiques, très conviviaux, vous permettent de réaliser nombre d'opérations sans taper une seule ligne de programme.

11. Commandes de bases

11.1. Format général des commandes

Une fois le système démarré, il attend les ordres que vous voudrez bien lui donner, sous la forme de ce que l'on appelle un **prompt**. Celui-ci qui peut présenter différentes formes selon les cas, par exemple:

```
[root@tchana ~]#
```

Dans celui-ci, les informations affichées sont (*de gauche à droite*):

- le nom de l'utilisateur;
- le nom de la machine sur laquelle cet utilisateur est connecté;
- le répertoire courant.

Le caractère ▮ symbolise le curseur. C'est à cet endroit que vous entrez les commandes au clavier, en terminant par la touche [**Entrée**] (*en général, la plus grosse du clavier, sur la droite des lettres*). Par entrer une commande, on entend frapper au clavier une suite de lettres ou autres caractères, de manière à constituer des mots que le système pourra interpréter. Dans l'exemple qui suit, le symbole "␣"représente un espace frappé au clavier (*qui donc n'affiche rien à l'écran*):

```
[root@tchana ~]# gzip␣-9␣Rapport.lyx␣Lettre.txt_
```

Les espaces sont très importants lorsque vous donnez une commande. Par exemple, les deux mots "**gzip␣-9**" seront interprétés de manière totalement différente par le système sous la forme "**gzip-9**", sans espace. Ceci fait

partie des règles à respecter pour donner correctement une commande au système; l'ensemble de ces règles est désigné sous le terme de syntaxe, qui a en fait le même sens que lorsqu'on l'emploie pour parler de la syntaxe de la langue française.

Par ailleurs, il n'est pas rare que les commandes produisent un résultat à l'écran, c'est-à-dire affichent diverses informations. Dans la suite, nous parlerons alors d'affichage résultant, pour désigner ce qu'une commande pourra ainsi afficher.

Normalement, le système exécute la commande (*si celle-ci est correctement orthographiée*) avant d'autoriser la saisie d'une nouvelle commande. Toutefois, Linux est multitâche: il est donc possible de lancer une commande en arrière-plan, et de récupérer immédiatement la main. La commande s'exécutera alors pendant que vous faites autre chose. Ceci est particulièrement intéressant pour les commandes de longue durée, dont on peut ignorer l'affichage résultant. On réalise l'exécution d'une commande en arrière-plan en la faisant suivre du symbole "et-commercial" '**&**'. Par exemple:

```
[root@tchana ~]# ls -alR / > tous_fichiers &
```

Cette commande va donner une liste complète des fichiers du système, et stocker le résultat dans le fichier **tous_fichiers**. C'est généralement quelque chose d'assez long (*un système Linux comporte souvent quelque dix mille ou vingt mille fichiers*)... Des explications permettant de comprendre cette commande viennent ci-dessous.

Comme vous l'avez constaté, nous utilisons une forme de caractère différente lorsqu'il s'agit de commandes ou de nom de fichiers.

11.2. Options

La plupart des commandes acceptent des options (*appelées aussi **paramètres***), qui permettent de préciser la manière dont elles doivent s'exécuter.

Sous Linux, la manière usuelle de spécifier des options est un tiret '**-**' précédé d'un **espace** et suivi d'une **lettre**, ou alors d'un double-tiret '**--**' suivi d'un **mot**. Il est généralement possible de combiner plusieurs options "monolettres" en une seule. Par exemple:

```
ls␣-lisa␣-color=tty (équivaut à ls␣-l␣-i␣-s␣-a␣-color=tty)
```

```
rpm␣-i␣libg++-2.7.2.8-9.i386.rpm␣-nodeps
```

Encore une fois, les espaces sont très importants. Dans la deuxième commande, le mot "**libg++-2.7.2.8-9.i386.rpm**" est un nom de fichier. Remarquez que ce nom comporte justement un tiret: l'absence d'espace avant permet au système de comprendre que ce qui vient après le tiret n'est pas une option, mais la suite du nom de fichier. Deux options très répandues sont:

```
-h
```

```
--help
```

Donner une (*courte!*) aide sur l'utilisation de la commande.

```
commande -V
```

Donner le numéro de version de la commande utilisée.

```
commande -ver
```


11.3. Redirections

Il est fréquent que les commandes produisent un résultat à l'écran ou parfois demandent des informations complémentaires à l'utilisateur. Il est possible de rediriger ces entrées/sorties à l'aide des symboles '>' et '<': par exemple, le résultat d'une commande est trop long pour être affiché dans l'écran, ou bien on désire en garder une trace. Une redirection de la sortie de cette commande permet de stocker ce qu'elle afficherait normalement dans un fichier.

La commande suivante liste tous les éléments (*habituellement désignés sous le vocable de packages*) d'un système installé à partir d'une distribution **RedHat**, **CentOS**, **NethServer** ou **SuSE**:

```
rpm -qa
```

Si vous possédez une telle distribution, le nombre de packages dépasse en général largement la hauteur de l'écran... Et c'est assez malcommode à consulter. Ce problème peut être résolu en envoyant l'affichage dans un fichier, qu'il sera alors possible de consulter à loisir ou d'imprimer:

```
rpm -qa > tous_les_packages.txt
```

Les redirections peuvent être effectuées dans l'autre sens. Supposons que vous souhaitiez envoyer un courrier électronique dont le texte se trouve dans un fichier. Une méthode consiste à exécuter:

```
mail untel@tel.adresse < message.txt
```

Ces exemples simples ne montrent pas toute la puissance des redirections. Disons simplement qu'il est possible de combiner les deux types ensemble, ou encore d'utiliser les pipes exposés maintenant...

11.4. Pipe

Ce mécanisme de **Pipe**[1] permet d'enchaîner l'exécution de plusieurs commandes, en branchant la sortie d'une commande sur l'entrée de la suivante, à l'aide du symbole '|' (*obtenu sur un clavier Français (Canada) usuel en pressant simultanément sur la touche **AltCar** et la touche à gauche du caractère **1***). Prenons un exemple.

Vous souhaitez connaître tous les fichiers de votre système dont le nom comporte les caractères '**doc**'.

La commande permettant d'obtenir tous les fichiers du système est '**ls -alR /**', nous l'avons déjà rencontrée.

Il existe une commande permettant de rechercher les lignes comportant un certain texte dans un fichier, et de n'afficher que celles-ci: la commande **grep** (*inutile pour l'instant de la comprendre pleinement!*).

Avec les redirections, nous pourrions obtenir ce que nous voulons par:

```
ls -alR / > tous_fichiers
```

```
grep doc tous_fichiers
```

Un pipe nous permet toutefois de simplifier l'expression et d'éviter la création d'un fichier:

```
ls -alR / | grep doc
```

1 **Pipe**: En français, on parle de tubes.

Le fonctionnement peut être illustré par la *Figure 2*.

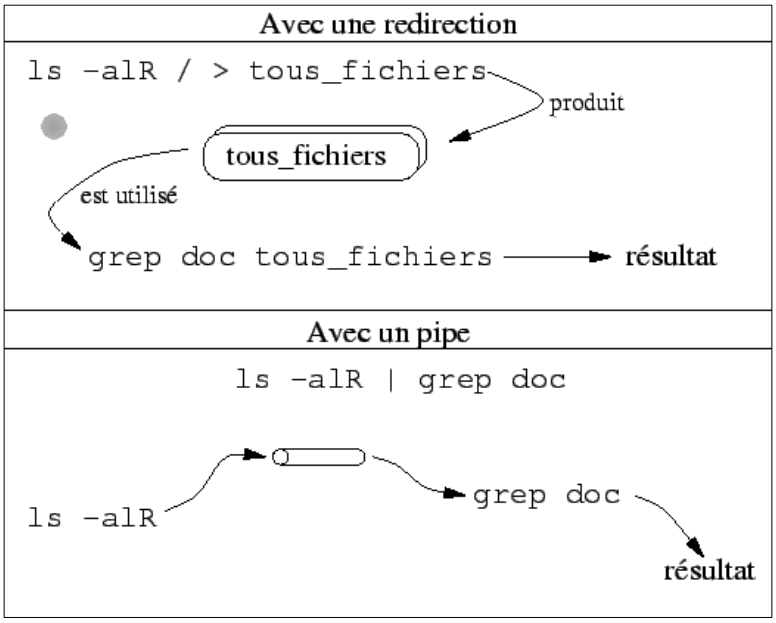


Figure 2: Mécanisme d'un pipe.

Ce qui se passe en réalité est ce qui est normalement affiché par la première commande, est utilisé par la deuxième comme s'il s'agissait d'un véritable fichier.

La combinaison des pipes et des redirections, en donnant les options judicieuses aux commandes impliquées, permet de réaliser en une seule ligne de commande des opérations très complexes. Mais gare à ne pas se perdre soi-même...

11.5. Jokers

Il est fréquent qu'une commande fasse référence à, ou utilise, un fichier. On peut toutefois vouloir appliquer une commande à plusieurs fichiers, parfois en très grand nombre; le fait de retaper la commande pour chaque fichier peut devenir rapidement fastidieux.

C'est pourquoi il existe les caractères **jokers**. Leur rôle est de remplacer une ou plusieurs lettres dans un nom de fichier, de la même manière que le joker d'un jeu de cartes remplace la carte manquante. Il existe deux caractères **jokers** selon ce que l'on veut faire:

Sans doute le plus utilisé, permet de remplacer un groupe de caractères (*lettres ou autre*).

```
*
```

Permet de remplacer un seul caractère.

```
?
```

Envisageons quelques exemples.

Vous souhaitez obtenir la liste de tous les fichiers du répertoire courant commençants par les lettres "**courrier_**". La commande est la suivante:

```
ls courrier_*
```


Le caractère '*****' est interprété comme "**n'importe quoi**": on demande donc la liste des fichiers commençant par "**courrier_**" suivi de "**n'importe quoi**". Par contre, la commande:

```
ls courrier_?
```

limite l'affichage aux fichiers dont le nom commence par "**courrier_**" suivi d'un seul caractère (*quel qu'il soit*).

Pour reprendre en partie l'exemple donné sur les pipes plus haut, une recherche équivalente dans le répertoire courant s'écrirait:

```
ls *doc*
```

12. man

Besoin d'un renseignement sur la fonction ou l'utilisation d'une commande? Des questions sur la structure d'un fichier de configuration? La commande **man** est là pour ça!

La commande **man** donne accès aux "**pages man**[2]", les pages de manuel. Ces pages se trouvent dans le répertoire **/usr/man**, mais peuvent également se trouver en d'autres endroits plus spécialisés (*voir* ***Figure 1***).

Ce que l'on appelle communément "le **man**" est subdivisé en plusieurs sections selon le propos des pages[3], comme le montre le ***Tableau 2***.

Nom	Description
1	Commandes utilisateurs usuels et communes
2	Appels système (*fonctions du kernel*)
3	Sous-programmes (*fonctions des librairies de programmation*)
4	Périphériques (*fichiers dans /dev*)
5	Format (*structure*) de certains fichiers, comme /etc/passwd
6	Jeux
7	Divers
8	Outils d'administration système (*réservé à root*)
9	Autres fonctions du noyau (*kernel*)
n	Documentation nouvelle (*qui peut être déplacée par la suite*)
o	Documentation ancienne (*qui peut disparaître un jour*)
l	Documentation locale (*spécifique au système que l'on utilise*)

Tableau 2: Les sections des pages man.

Pour accéder à la page concernant une commande (*par exemple, **chmod***), il suffit de taper:

```
man chmod
```

La commande **man** possède de nombreuses options de ligne de commande. Citons par exemple:

Permettre d'obtenir toutes les pages concernant le nom donné (*essayez par exemple **man -a kill***).

```
man -a kill
```

2 **man**: man-pages en anglais, "man" pour manual, manuel en français.
3 **pages**: Liste extraite du Man-Page mini-Howto.

Pour provoquer un formatage PostScript de la page, avant de l'envoyer sur la sortie standard *(-t)*.

Une redirection permet d'obtenir un fichier PostScript immédiatement imprimable:

```
man -t adduser > adduser.ps
```

Il existe de nombreuses autres possibilités, mais rien de vous empêche de faire:

```
man man
```

Enfin, si vous savez dans quelle section se trouve la page cherchée (*dans le cas de commandes possédant un équivalent en programmation*), il est possible d'indiquer la section où chercher la page voulue:

```
man 2 mount
```

```
man 8 mount
```

Le **man** est une source inépuisable de renseignements. Surtout ne jamais le négliger! Mais il est également possible d'obtenir de nombreuses informations dans le répertoire **/usr/doc**, notamment le sous-répertoire **HOWTO**.

13. Parcours et manipulation des répertoires

13.1. Promenade dans les répertoires

La commande **cd** permet de changer le répertoire courant.

Supposons nous trouver dans le répertoire **/home**, et pour quelque raison nous voulons aller dans le répertoire **/usr/doc**. Les trois séries de commandes suivantes effectuent l'opération souhaitée:

```
cd /usr/doc
```

```
cd ../usr/doc
```

```
cd ..
cd usr
cd doc
```

Encore un exemple: nous sommes dans **/usr/local/doc**, et nous voulons aller dans **/usr/X11R6/doc**. Trois manières d'y parvenir:

```
cd ..
cd ..
cd X11R6
cd doc
```

```
cd ../../X11R6/doc
```

```
cd /usr/X11R6/doc
```

A vous de trouver laquelle vous préférez...

13.2. Création de répertoires

mkdir est utilisée pour créer un répertoire. Il est naturellement possible d'utiliser un chemin relatif ou un chemin absolu. Pour créer un répertoire dans le répertoire courant, il suffit de taper quelque chose du genre:

```
mkdir rep
```


Une option intéressante est l'option '**-p**': elle permet de créer en une fois une hiérarchie de répertoire. Supposons que dans le répertoire courant, le répertoire **Lettres** n'existe pas, que l'on veuille le créer, ainsi qu'un sous-répertoire **perso**, contenant lui-même un répertoire **Nath**. On peut le faire en exécutant simplement:

```
mkdir -p Lettres/perso/Nath
```

13.3. Suppression de répertoires

La commande **rmdir** est utilisée pour supprimer un répertoire. Normalement, si le répertoire n'est pas vide, la commande envoie un message d'erreur. Il est toutefois possible de supprimer une hiérarchie de répertoires vides en une seule fois, avec l'option '**-p**'. Par exemple, juste après avoir exécuté:

```
mkdir -p Lettre/perso/Nath
```

on peut supprimer tous les répertoires que l'on vient de créer avec:

```
rmdir -p Lettre/perso/Nath
```

Par contre, la commande

```
rmdir -p Lettre/perso
```

renverra une erreur, car **perso** contient encore **Nath** et donc n'est pas vide.

13.4. Savoir où l'on est

La commande **pwd** toute simple donne le répertoire courant, par son chemin absolu. Bien pratique dans certains cas...

13.5. Encombrements des répertoires et des fichiers

Deux outils pour le contrôle de l'espace disque occupé par les fichiers.

du permet de connaître l'espace disque occupé par les fichiers d'un répertoire.

df permet de connaître l'espace disque total occupé et restant sur les partitions de votre système.

13.6. Illustration

La ***Figure 3*** illustre l'utilisation des commandes évoquées dans cette section. Le répertoire courant apparaît en grisé sur le dessin. Naturellement, seule une partie de l'arborescence est présentée.

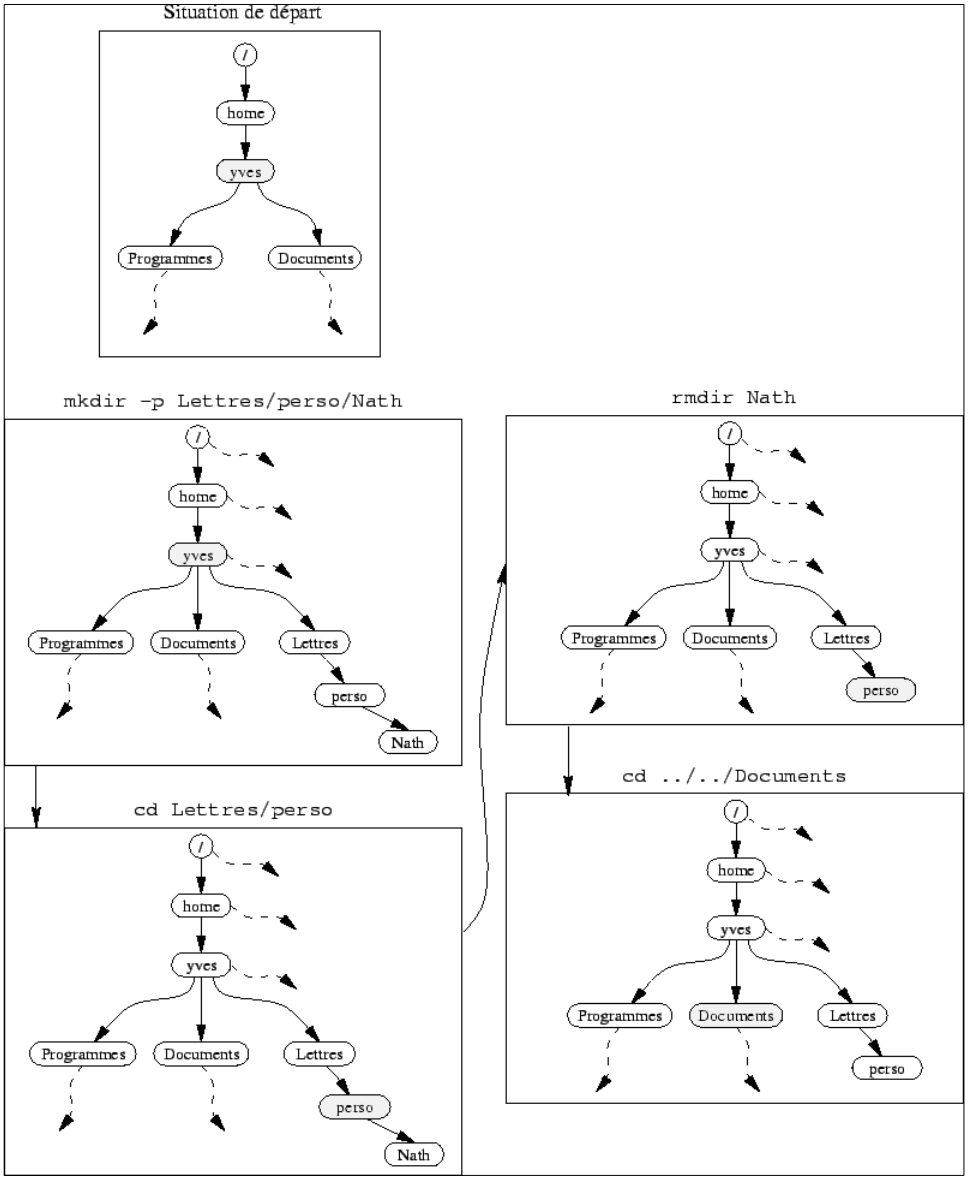


Figure 3: Démonstration de manipulations de répertoires.

14. Manipulation des fichiers

14.1. Liste des fichiers dans un répertoire

ls donne la liste des fichiers contenus dans un répertoire. Appelée sans arguments, **ls** donne la liste des fichiers dans le répertoire courant.

Il existe de nombreuses options à **ls**. Nous n'évoquerons ici que les principales et les plus utiles, pour le reste, consultez la page **man** correspondante ('***man ls***').

Lister les fichiers cachés (*dont le nom commence par un point*).

```
ls -a
```

Affichage "**long**": donne des informations supplémentaires sur les fichiers, tels que la taille, la date de dernière modification, les droits d'accès, etc.

```
ls -l
```

Lister récursivement le contenu des sous-répertoires. Essayez donc un '**ls -alR /**'...

```
ls -R
```

Affiche les fichiers en les ordonnant selon leur extension (*les caractères après le dernier point*).

```
ls -X
```

Lister sur une seule colonne. Par défaut, **ls** utilise plusieurs colonnes, selon la longueur des noms à afficher.

```
ls -1
```

Appliquer une couleur selon le type de fichier: une pour les répertoires, une pour les commandes exécutables, etc.

```
ls -color=tty
```

14.2. Supprimer des fichiers

Il est évidemment possible de supprimer des fichiers sous Linux. La commande à utiliser est **rm**. Prenez garde aux erreurs de manipulation: une fois un fichier effacé, il n'est plus possible de récupérer son contenu, celui-ci est irrémédiablement perdu. Par exemple, s'il existe dans le répertoire courant un fichier nommé **core** (*les fichiers de ce nom sont généralement générés lorsqu'un programme provoque une erreur*), vous pouvez le supprimer par:

```
rm core
```

14.2.1. Les options les plus utilisées de rm

Supprimer tout un répertoire, ainsi que ses sous-répertoires.

Attention! Sur certains systèmes, une commande telle que '**rm -R .***' peut avoir des conséquences catastrophiques: elle supprime le répertoire courant, ainsi que les répertoires qu'il contient. Or, le répertoire '**..**', qui désigne l'ancêtre du répertoire courant, est contenu dans celui-ci: sur certaines versions, **rm** va alors "remonter" dans l'arborescence, parvenir à la racine, puis redescendre... En quelques secondes, vous pouvez ainsi supprimer tous les fichiers de votre système Linux!

```
rm -R
```

Demander une confirmation avant de supprimer un fichier.

```
rm -i
```


14.3. Copier des fichiers

La copie de fichiers (*c'est-à-dire une duplication à l'identique, comme un clonage*) s'effectue par **cp**. Il est toujours nécessaire d'indiquer ce que l'on copie (*la source*), puis vers quel endroit on le copie (*la destination*). Les caractères **jockers** (voir le paragraphe ***Jokers*** à la page ***18***) peuvent bien entendu être utilisés.

Méfiez-vous lorsque vous utilisez cette commande, surtout en tant qu'utilisateur **root** (voir le paragraphe ***Le super-utilisateur*** à la page ***43***): sauf configuration particulière, si vous copiez un fichier sur un autre de même nom dans un autre répertoire, ce dernier sera purement et simplement écrasé sans avertissement, c'est-à-dire que son contenu sera irrémédiablement perdu.

Enfin, la commande **cp** peut être utilisée pour simplement dupliquer un fichier dans un même répertoire. Par exemple, si dans le répertoire courant se trouve un fichier nommé **Rapport.lyx**, que vous souhaitez modifier mais en conservant la version précédente, vous pouvez effectuer:

```
cp Rapport.lyx Rapport.lyx.org
```

Le fichier actuel sera ainsi sauvegardé en tant que **Rapport.lyx.org**. Si vous n'êtes pas satisfait de vos modifications, vous pourrez toujours restaurer la version précédente par:

```
cp Rapport.lyx.org Rapport.lyx
```

Nous utilisons ici le fait que copier un fichier sur un fichier de même nom, écrase celui-ci: le **Rapport.lyx** sera exactement identique à celui que vous aviez sauvegardé avec la première commande.

Une utilisation plus courante de **cp** est de copier un fichier situé quelque part dans le répertoire courant, pour par exemple l'éditer. Si vous souhaitez ainsi disposer dans le répertoire courant du fichier **Printing-HOWTO** situé dans le répertoire **/usr/doc/HOWTO**, vous pouvez utiliser:

```
cp /usr/doc/HOWTO/Printing-HOWTO .
```

Le '**.**' final désigne, rappelez-vous, le répertoire courant.

14.3.1. Quelques options

Pour copier une arborescence récursivement: si la source est un répertoire, cette option va copier ce répertoire et son contenu ainsi que tous les sous-répertoires qu'il contient.

```
cp -R source destination
```

Demander une confirmation si un fichier risque d'être écrasé.

```
cp -i source destination
```

Afficher le nom de chaque fichier avant de le copier. Utile lorsqu'on copie un répertoire ou qu'on utilise les caractères **jockers**.

```
cp -v source destination
```

14.4. Déplacer ou renommer des fichiers

mv permet de déplacer des fichiers (*les fichiers sont copiés comme avec **cp**, puis effacés de la source*). Si le répertoire source est le même que le répertoire destination, cette commande permet de renommer des fichiers.

Par exemple, s'il se trouve dans le répertoire courant un fichier nommé **Lettre** que vous voudriez renommer (*changer son nom*) en **Lettre-2017.01.06** (*une bonne manière d'insérer la date dans un nom de fichier*), vous pouvez effectuer:

```
mv  Lettre  Lettre-2017.01.06
```


Supposons maintenant que vous souhaitiez déplacer ce fichier dans un répertoire qui s'appellerait **Archives**, c'est-à-dire placer une copie du fichier dans ce répertoire et supprimer l'original. Une première méthode est:

```
cp Lettre-2017.01.06 Archives
```

```
rm Lettre-2017.01.06
```

La commande **mv** permet de faire la même chose en une seule ligne:

```
mv Lettre-2017.01.06 Archives
```

De même qu'avec la commande **cp**, méfiez-vous; cette commande peut écraser des fichiers si la cible existe déjà.

mv possède peu d'options.

Forcer le déplacement, même si la cible existe déjà: ceci supprime les éventuels messages d'avertissement.

```
mv -f source destination
```

Au contraire, l'option **-i** provoque une demande de confirmation si des fichiers risquent d'être écrasés.

```
mv -i source destination
```

14.5. La complétion des noms de fichier

Il s'agit là d'un petit quelque chose très pratique du **shell Bash**. Prenons un exemple. Entrez la commande suivante (*sans danger*):

```
cd /etc
```

Le résultat de cette commande est de nous amener dans le répertoire **/etc**, qui devient donc le répertoire courant.

Maintenant, tapez les caractères suivants, mais n'appuyez pas immédiatement sur [**Entrée**]:

```
more H
```

Appuyez maintenant sur la touche de [**Tabulation**]. Normalement, vous devriez voir apparaître:

```
more HOSTNAME
```

Que s'est-il passe? Lorsque vous avez appuyé sur la touche [**Tabulation**], le shell a recherché dans le répertoire courant les fichiers dont le nom commence par la lettre '**H**' (*rappelez-vous que Linux distingue les majuscules des minuscules dans les noms de fichiers*). En principe, il n'y en qu'un seul, et le shell complète alors le nom du fichier. Recommencez, avec cette fois:

```
more f
```

suivi à nouveau de la touche [**Tabulation**]. Rien ne se passe, avec un peu de chance vous devriez même entendre un **bip**: c'est que plusieurs fichiers commencent par la lettre '**f**'. Appuyez une deuxième fois sur [**Tabulation**] rapidement, et le **shell** affichera tous les fichiers commençant par '**f**'.

Ceci fonctionne également si vous donnez un groupe de lettres, ou un chemin (*relatif ou absolu*).

Si [**Tabulation**] est suivi d'un **bip**, plusieurs fichiers commencent par '**rc**' dans le répertoire **/etc/rc.d**.

```
more /etc/rc.d/rc
```


Cette petite fonctionnalité est très pratique dès que l'on se trouve confronté à des noms de fichiers un peu longs. Expérimentez-la, vous l'apprécierez rapidement!

Pour vous replacer dans votre répertoire de login.

```
cd
```

14.6. Affichage de fichiers (*texte*)

more et **less** permettent de consulter le contenu d'un fichier à l'écran, en l'affichant page par page. Par exemple:

```
more /etc/inittab
```

```
less /etc/passwd
```

Vous pouvez passer à la page suivante en appuyant sur la **barre d'espace** de votre clavier. Pour remonter d'une page avec **less**, il suffit de presser la touche '**b**' (*back*). Normalement, **less** ne vous rend pas automatiquement la main: une fois arrivé à la fin du document, il faut presser la touche '**q**' pour terminer **less** et pouvoir entrer une autre commande.

La commande **more** est plus ancienne. La commande **less** est une sorte de "**super-more**", en ceci qu'elle ajoute nombre de fonctionnalités. Consultez la page **man** correspondante pour plus de détails. **less** accepte des commandes lors de son fonctionnement: vous les tapez simplement au clavier lorsque **less** a fini d'afficher une page, suivies d'un appui sur [**Entrée**].

Effectuer une recherche du motif dans le texte (*les caractères '<' et '>' ne doivent pas être inclus*). Très utile pour rechercher un mot dans un document long: pour rechercher le mot "**bonjour**", entrez la commande **/<bonjour>**.

```
/<motif>
```

Retourner au début du fichier.

```
g
```

Aller à la fin du fichier.

```
G
```

Aller à la ligne **n** du fichier.

```
ng
```

Il est assez fréquent d'utiliser ces commandes à travers un pipe (voir ***Pipe*** à la page ***17***) pour lire un affichage long résultant d'une commande. Par exemple, la commande **dmesg** permet de voir les messages du système lors du démarrage: cela représente souvent plus d'une page-écran. Pour pouvoir consulter facilement ces messages, vous pouvez effectuer:

```
dmesg | more
```


14.6.1. Illustration

La *Figure 4* vous propose une démonstration des commandes **rm**, **cp** et **mv** évoquées plus haut. Les répertoires sont dans des cadres aux coins arrondis, les fichiers dans des cadres aux coins droits. Le répertoire courant est toujours **yves**.

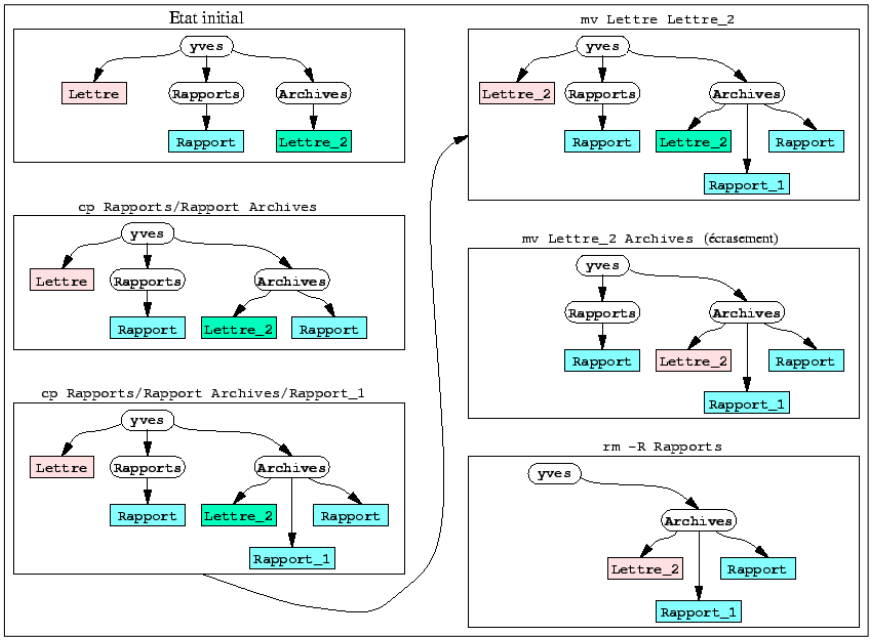


Figure 4: Démonstration de manipulation de fichiers.

Le contenu des fichiers est symbolisé par un motif de remplissage: un même motif remplissant les cadres de deux fichiers différents, signifie des contenus identiques.

15. Programme, processus, logiciel & Co

Nous allons tenter ici de clarifier quelques termes d'emploi fréquent. Commençons par un que nous avons déjà rencontré, le terme de **programme**.

Une définition un peu abstraite, tirée d'un dictionnaire, est de dire qu'un **programme** est une suite d'instructions compréhensible par l'ordinateur, organisée de manière à ce que celui-ci accomplisse une tâche donnée. C'est sans doute exact, mais ça ne nous renseigne pas sur la forme que prennent les **programmes** sur notre système Linux...

Essayez donc la commande suivante:

```
more /bin/more
```

Normalement, vous devriez voir une série de caractères parfaitement incompréhensible, peut-être même entendre quelques **beeps** et récupérer un affichage désordonné (*dans ce dernier cas, changez de console virtuelle, comme indiqué au paragraphe Notion de console à la page 14*). Que vient-il de se passer?

La commande **more** vous est déjà connue, elle permet d'afficher le contenu d'un fichier. Lorsque vous appuyez sur [**Entrée**] après avoir tapé la commande, vous demandez au système d'exécuter un programme; **more** en l'occurrence. Mais sans doute avez-vous remarqué que le fichier dont nous avons demandé l'affichage porte le même nom que la commande...

Ce fichier est justement celui qui contient le programme correspondant à la commande **more**. Les instructions d'un programme (*ce qu'il doit faire*) sont en effet contenues dans un fichier, le plus souvent sous une forme que l'œil humain ne peut saisir mais, aisément utilisable par l'ordinateur (*on parle alors de fichier binaire et tous les programmes ne se présentent pas sous cette forme*). La très grande majorité des commandes sont donc des programmes dont les instructions sont contenues dans des **fichiers binaires**.

Mais continuons notre enquête. Lorsque le système tente d'exécuter un programme, il lit le fichier qui le contient pour le placer dans sa mémoire. Alors seulement les instructions du programme sont exécutées, laquelle exécution se traduit le plus souvent par l'utilisation de davantage de mémoire par la lecture ou l'écriture de divers fichiers, des affichages à l'écran... Le programme que nous avons exécuté plus haut, par exemple, a lu un fichier (***/bin/more***) et affiché son contenu à l'écran.

On se trouve alors en présence d'une entité dans la mémoire de l'ordinateur, qui évolue, qui agit sur différents éléments qui lui sont accessibles. On pourrait même considérer qu'elle est née (*lors de la lecture du "**fichier-programme**" par le système*), qu'elle vit (*lors de l'exécution des instructions contenues dans le programme*), et qu'elle mourra (*lorsque sa tâche sera accomplie*). Cette entité presque vivante est ce que l'on appelle un **processus**. Dans un système tel que Linux, tout processus est "lancé" par un autre processus et est à même d'"enfanter" un ou plusieurs processus: on peut donc parler de "**relation père-fils**" entre les processus. Sachez qu'il est très courant d'avoir plus d'une vingtaine de processus s'exécutant simultanément lors du fonctionnement d'un système Linux, éventuellement communiquant entre eux. Pour s'y retrouver, le système attribue à chacun d'eux un numéro unique, désigné par le terme de **PID** (**P**rocess **Id**entifier[4]). Par ailleurs, un même programme peut être exécuté plusieurs fois: à chaque exécution correspondra un processus à part.

Naturellement, la plupart des processus ne font guère parler d'eux: sinon, votre écran serait rapidement rempli de messages et il vous serait impossible de vous retrouver dans un fouillis pareil. En fait, la plupart des **processus** tournent en arrière-plan; c'est-à-dire qu'ils font ce qu'ils doivent faire en silence. Par exemple, le **processus** qui vous permet d'accéder au système n'intervient que lorsqu'il est sollicité; le reste du temps il est silencieux. On parle communément de **daemons**[5] pour désigner de tels **processus** lorsqu'ils font partie des composantes du système.

Pour finir notre "bestiaire occulte", citons le cas des **zombies**. Ce sont des **processus** parvenus au terme de leur exécution (*soit normalement, soit à la suite d'une erreur*), mais que le système ne parvient pas à éradiquer de la mémoire. Ils demeurent donc ainsi, le plus souvent totalement inactifs, mais occupant une certaine quantité de mémoire, ce qui peut devenir gênant si leur nombre augmente trop. En général, cet état survient quand un **processus** se termine alors que le système n'a pas le temps de s'occuper de l'effacer de la mémoire parce qu'il est trop sollicité. Cela peut également arriver lorsque le **processus** a provoqué une erreur telle que le système ne sait pas quoi en faire. Mais cela reste assez rare, ne vous inquiétez pas.

Nous savons maintenant ce qu'est un programme et un **processus**. Donnons alors quelques définitions.

Un **logiciel** est un ensemble de programmes fortement liés entre eux, destinés à remplir une mission nécessitant plusieurs tâches plus ou moins complexes. Par exemple, un traitement de texte qui peut aussi bien être utilisé pour rédiger un curriculum vitae que pour rédiger un livre entier.

Un **système d'exploitation**, dont nous parlons depuis le début, est un logiciel particulier; son rôle est de permettre à d'autres programmes de s'exécuter dans les meilleures conditions et d'utiliser le matériel électronique de l'ordinateur. L'exécution d'un système tel que Linux peut donner lieu à l'apparition d'une multitude de processus.

Enfin, ce que l'on désigne par **code source** est ce qu'un (*ou plusieurs*) être humain a tapé au clavier, dans un certain langage de programmation compréhensible par lui, pour construire un programme. Ce texte n'est pas compréhensible par l'ordinateur et doit subir une phase de traduction (*ou compilation*) pour obtenir un **fichier binaire**. Par exemple, Linux lui-même est programmé dans un langage (*presque*) intelligible, nommé langage **C** et normalement le code source de Linux se trouve dans une arborescence située dans le répertoire **/usr/src/linux**. Notez qu'habituellement le **code source** des logiciels et systèmes commerciaux n'est pas ainsi librement

4 Identifiant ou Identificateur du Processus en français.

5 **daemon**: De l'anglais daemon **D**isk **a**nd **E**xecution **Mon**itor. La traduction n'est peut-être pas très heureuse linguistiquement, mais elle correspond bien à l'idée.

disponible: Linux (*et la plupart des outils qui l'accompagnent*) appartient au mouvement **LIBRE** (*OpenSource*) et fait figure d'exception.

16. Programmes usuels

16.1. Manipuler fichiers et répertoires aisément: mc

Ce petit programme est inspiré de **Norton Commander** qui était disponible pour le système MS-DOS. Son nom "complet" est **Midnight Commander** et on l'obtient avec la commande **mc**. Il en résulte alors un écran ressemblant à celui ci-contre.

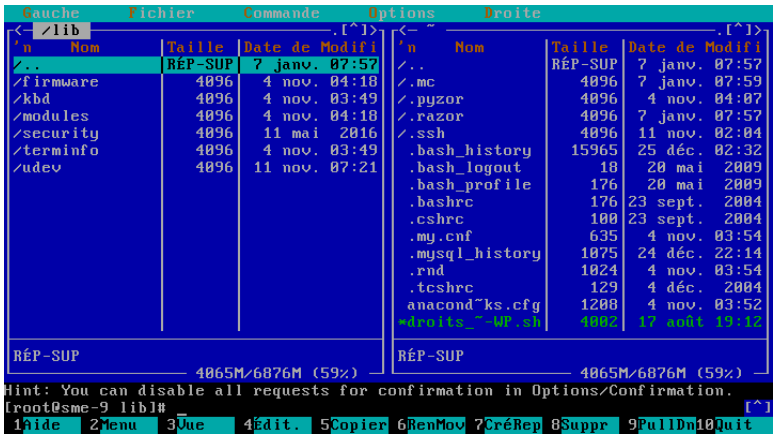


Figure 5: Écran de Midnight Commander (mc).

Il est possible que le résultat que vous obtenez ne soit pas en couleur: il faut alors utiliser l'option **-c** et exécuter plutôt "mc -c".

L'écran se trouve divisé en deux parties que nous appellerons les volets de gauche et de droite. Chacun vous permet de consulter le contenu d'un répertoire dont le chemin complet figure en haut du volet. Dans notre exemple, le volet de gauche affiche le contenu du répertoire **/lib**, tandis que celui de droite affiche le contenu du répertoire propre à l'utilisateur qui a lancé **mc** (*c'est le sens du symbole "~", qui représente toujours le répertoire principal d'un utilisateur*).

Dans les listes contenues dans chacun des volets, différents codes de couleur sont utilisés pour représenter différents éléments. Ainsi, les répertoires contenus dans le répertoire visité apparaissent en blanc brillant et caractères gras, précédés d'un slash ('/'). Les fichiers exécutables (*correspondant à des programmes*) apparaissent en vert et le nom est précédé d'une étoile ('*****'). Les fichiers dont le nom est précédé d'une arobase ('**@**') sont des liens symboliques (*voyez le paragraphe* ***Mécanisme des liens*** *à la page* ***42*** *pour plus d'information sur les liens*).

Sur la droite des noms de fichier, vous voyez également une colonne avec la taille du fichier (*en octets*) et une colonne avec la date de dernière modification du fichier. Dans l'un des deux volets, il y a toujours un fichier sélectionné et celui-ci est marqué par une barre de couleur; dans notre exemple c'est le répertoire **/..** du volet de gauche qui est sélectionné. Vous pouvez déplacer la sélection avec les touches de déplacement (*flèches* ***Haut*** *et* ***Bas***), ou bien les touches [**Début**] (*retour au début de la liste*), [**Fin**] (*aller à la fin de la liste*), [**Page Avant**] (*descendre de plusieurs lignes*) et [**Page Arrière**] (*remonter de plusieurs lignes*). Pour passer d'un volet à l'autre, utilisez la touche de [**Tabulation**].

Tout en bas de l'écran figure la liste des fonctions que vous pouvez appeler, à l'aide d'une des touches de fonction. Le fonctionnement est assez simple. Par exemple, si vous pressez la touche [**F5**] (*notez le mot* ***"Copy"*** *à côté du* ***5*** *dans la ligne du bas*), cela va réaliser la copie de l'élément sélectionné dans un volet vers le répertoire affiché dans l'autre volet. Vous pouvez ainsi copier un seul fichier, ou bien toute une arborescence.

Essayez-le, vous ne tarderez pas à l'apprécier!

III- Quincaillerie

1. Notions d'architecture matérielle

Linux demeure un système exigeant. Si vous devez "administrer" une machine Linux (*la configurer, installer le système...*), nous pensons qu'il est indispensable de posséder quelques notions concernant les matériels des ordinateurs de type PC. Aussi aborderons-nous ici quelques aspects essentiels et volontairement simplifiés.

2. Les disquettes et les disques durs

Les disquettes et les disques durs entrent dans la catégorie des supports de mémoire de masse. Bien que cela ne soit pas tout à fait vrai pour une disquette, un disque dur est généralement capable de stocker beaucoup plus d'informations que la mémoire centrale d'un ordinateur. En général, le disque dur est le support sur lequel les systèmes d'exploitation, les logiciels divers que vous utilisez ainsi que vos fichiers personnels sont installés.

La disquette était souvent utilisée comme outil de sauvegarde ou de transport de données si celles-ci n'étaient pas trop volumineuses. Dans le cas de Linux ou MS-DOS, il était également possible d'installer un système d'exploitation sur une disquette, encore que ses fonctionnalités soient grandement réduites. Habituellement le lecteur de disquettes est le premier endroit où l'ordinateur va chercher un système lorsqu'il démarre; c'était souvent la seule façon de réparer un système sur disque dur qui avait été endommagé d'une manière ou d'une autre.

Physiquement, une disquette était un disque souple de plastique magnétisé comparable à une bande de cassette audio ou vidéo. Les informations étaient enregistrées sur ce support de façon analogue. Un disque dur se présente de la même façon, mais cette fois le disque est rigide, plus épais et se compose de plusieurs plateaux superposés. Voyez les photos de la ***Figure 6*** pour apprécier la façon dont un disque dur est construit. Le nombre de têtes de lecture, c'est-à-dire le nombre de faces utilisés sur les différents plateaux, est l'un des éléments de ce que l'on appelle la géométrie d'un disque dur. Pour une disquette, ce nombre valait 2 dans la grande majorité des cas (*un disque, donc deux faces, donc deux têtes*).

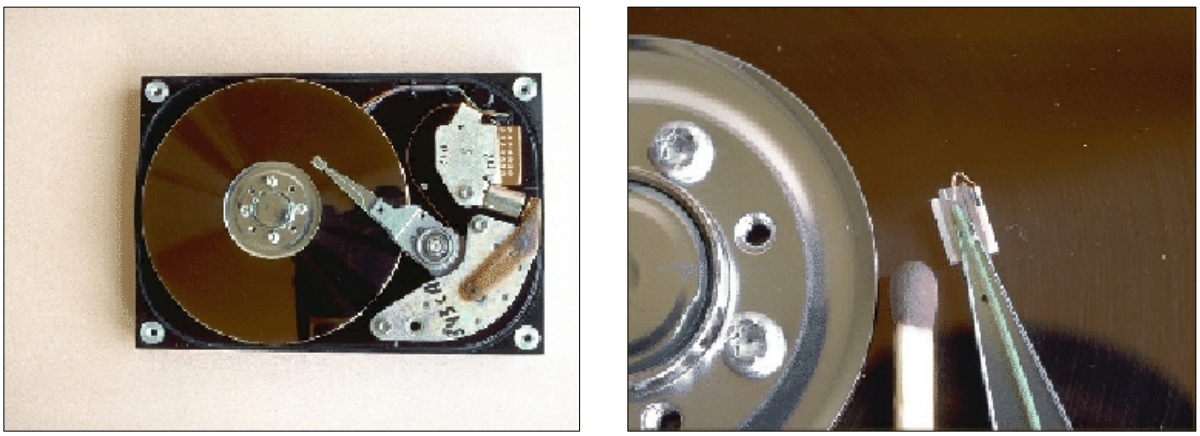


Figure 6: L'intérieur d'un disque dur.

Un autre élément est le nombre de **pistes** ou de **cylindres**. Les pistes sont des anneaux concentriques subdivisant la face d'un plateau en autant de zones distinctes. Sauf situation très exceptionnelle, toutes les faces du disque présentent le même nombre de pistes. Lorsqu'il y a plusieurs plateaux (*dans le cas des disques durs*), les pistes des différentes faces qui sont à l'aplomb les unes des autres forment un ensemble que l'on désigne alors sous le terme de **cylindre**. Une disquette usuelle comportait 80 cylindres, certains disques durs, plusieurs milliers.

Le "découpage" du disque ne s'arrête pas là. Les pistes sont elle-mêmes subdivisées en **secteurs**. Le secteur est la plus petite unité discernable sur un disque. Le plus souvent, toutes les pistes ont le même nombre de secteurs, mais il arrive que les pistes extérieures en possèdent davantage que les pistes intérieures pour tirer parti de la surface plus grande. Ceci est toutefois assez rare. Un secteur contient 512 octets ou caractères de données. D'autres tailles sont possibles (128, 256, 1024, 2048, 4096...). Une disquette normale contenait 18 secteurs par piste et un disque dur en contient au moins 63.

La ***Figure 7*** vous propose un schéma de la structure qui vient d'être évoquée. Sur la vue de dessus, les zones d'un même ton de gris appartiennent à la même piste. Sur la vue par la tranche (*imaginez le disque dur coupé en son centre*), les tons de gris représentent alors chacun un cylindre particulier.

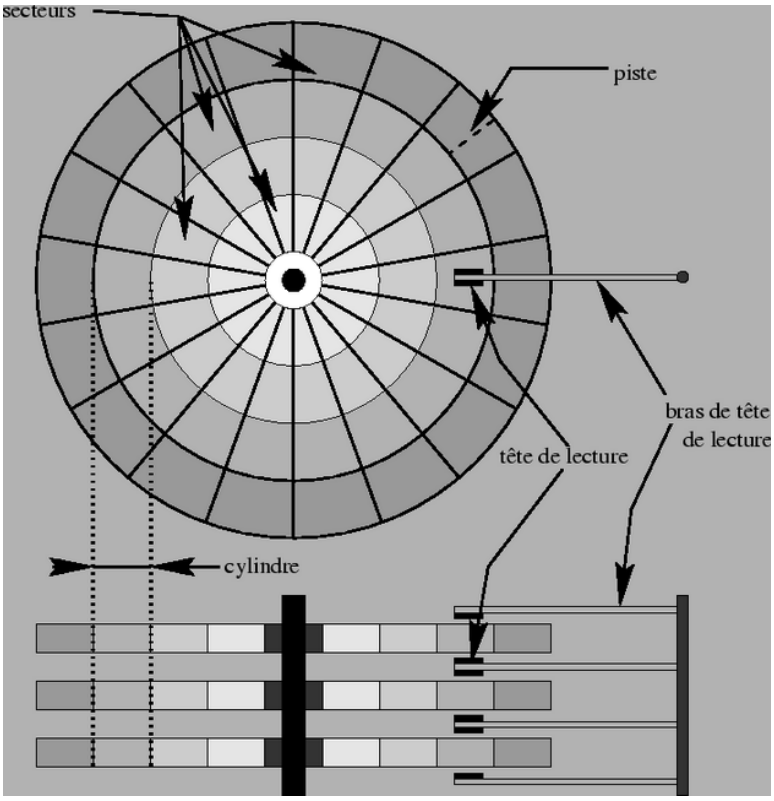


Figure 7: Structure d'un disque dur: vue de dessus et par la tranche.

En termes de performances, le taux de transfert maximal pour une disquette avoisinait les 300 kilo-octets par seconde tandis que certains disques durs atteignent des taux tels que 80 méga-octets par seconde, voire davantage! Un paramètre important est le temps moyen d'accès à une information; pour un disque dur, il est de l'ordre de quelques millisecondes (*millième de seconde*) et pour la disquette... certains esprits chagrins ironisent en disant qu'il fallait changer d'échelle et compter plutôt en minutes...

3. Partition des disques durs

Il existe un mécanisme qui permet de subdiviser un disque dur en plusieurs parties distinctes: les partitions. Celles-ci permettent de séparer, par exemple, les éléments constitutifs du système des données des utilisateurs. Ainsi, en cas de problème dans le système, les erreurs ont moins de chance de s'étendre aux données - et réciproquement. Typiquement, sur un système Linux, les répertoires qui contiennent les données des utilisateurs sont sur une (*ou plusieurs*) partition distincte du reste du système; une surcharge de données ne risque donc pas d'endommager le système et une erreur dans celui-ci a peu de chance de corrompre les données.

Le découpage d'un disque en partitions suit des règles un peu complexes, pour l'essentiel héritées de l'histoire. La première d'entre elles, est qu'une partition recouvre une suite continue de cylindres: les limites de partitions sont donc celles des cylindres et l'on dit qu'une partition s'étend du cylindre **n** au cylindre **m**. Une formule qui peut être utile donne la taille d'une telle partition, en méga-octets:

$$\frac{(m - n + 1) * Nt * Ns}{2048}$$

où ***Nt*** est le nombre de têtes, et ***Ns*** le nombre de secteurs par piste, du disque.

On distingue par ailleurs deux types de partitions: les partitions **primaires** et les partitions **logiques**.

Pour des raisons historiques, datant de l'époque où les disques étaient de taille bien moins importante qu'aujourd'hui, il n'était possible de créer que quatre partitions primaires qui sont numérotées par Linux de 1 à 4. Cette limite vient du fait que la faible capacité des disques durs ne justifiait pas de prévoir un nombre plus élevé.

Toutefois les capacités actuelles des disques rendent ce nombre insuffisant. C'est pourquoi fut développé la technique des partitions étendues et des partitions logiques.

Une partition étendue est l'une des quatre partitions primaires (*généralement la dernière*), destinée à être un réceptacle pour d'autres partitions en nombre théoriquement illimité. Ces partitions, contenues dans une partition étendue, sont appelées **partitions logiques**. Linux les numérote toujours à partir de 5.

Les partitions trouvent également leur utilité lorsque l'on veut faire cohabiter plusieurs systèmes. Il est possible, sur un même ordinateur, d'avoir côte à côte Windows-7, Windows-8.1, Linux... et même plusieurs fois le même système! Par exemple, un Linux d'utilisation normale et un autre pour effectuer des tests sur lequel un plantage est sans grande conséquence.

La manipulation des partitions se fait à l'aide d'outils spécialisés. Linux possède un programme nommé **fdisk**. Nous vous recommandons toutefois, si cela vous est possible, l'utilisation du programme **GParted** plus simple à manipuler et beaucoup plus puissant.

Les ***Figure 8*** et ***Figure 11*** vous propose deux exemples pour illustrer tout cela.

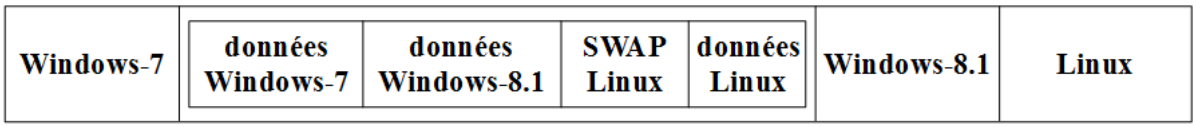


Figure 8: Exemple de partitions sur un disque dur (voir Tableau 3).

Dans cet exemple, les quatre possibilités de partition primaire ont été utilisées. L'une d'entre elles est utilisée en partition étendue (*la deuxième*), pour contenir des partitions logiques (*la deuxième partition primaire contient quatre partitions logiques*).

L'exemple suivant est plus classique.

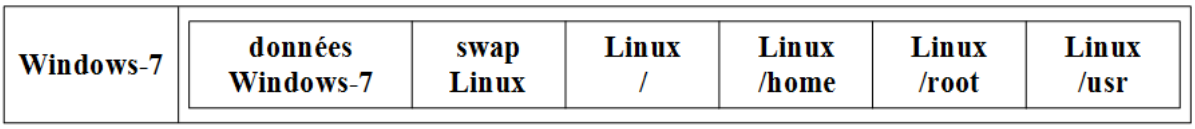


Figure 9: Exemple de partitions sur un disque dur.

Seules deux possibilités de partition primaire sont utilisées, la première pour Windows-7, la deuxième pour faire une partition étendue qui contient en l'occurrence six partitions logiques. Cet exemple vous montre par ailleurs comment il est possible de répartir différents éléments d'un système Linux sur plusieurs partitions: notez comment les répertoires **/home**, **/root** et **/usr** ne sont pas sur la même partition que la racine.

4. Structure logique d'un disque dur

Le début de cette section vous a exposé la structure physique d'un disque dur, divisé en cylindres, secteurs, etc. Puis nous avons évoqué le mécanisme des partitions. Mais où sont donc stockées les informations relatives aux partitions? Comment les systèmes reconnaissent-ils les partitions? Les réponses à ces questions sont l'objet de ce paragraphe.

La structure logique d'un disque dur est la manière dont sont organisées les différentes informations fondamentales concernant la manière dont le disque dur doit être utilisé par les systèmes.

Le premier secteur (*physique*) de chaque partition est un peu particulier: on l'appelle le **secteur d'amorce**[6], car il contient généralement le code nécessaire au lancement du système contenu dans la partition. S'il ne s'agit que d'une partition de données, ce secteur contient quelques informations utiles concernant la taille de la partition, son type, etc.

Encore plus particulier est le premier secteur physique du disque dur: ce secteur contient non seulement le code nécessaire au lancement du système actif sur votre système, mais en plus la fameuse **Table des Partitions**; cette table contient les paramètres des quatre partitions primaires du disque, c'est-à-dire les cylindres de début et de fin, le type, etc. Ce secteur si spécial, si fondamental pour le fonctionnement de la machine, est appelé le **Master Boot Record**[7], **MBR** en abrégé.

La **Table des Partitions** n'est toutefois pas suffisante en elle-même: elle ne peut contenir les informations des éventuelles partitions logiques contenue dans une ou plusieurs partitions étendues. Aussi, chaque partition logique contient une "mini" **Table des Partitions** pour réaliser en quelque sorte, un chaînage des partitions les unes au bout des autres. Le début de la chaîne, le premier maillon, se situe dans la **Table des Partitions** du **MBR**, où l'une des entrées est marquée comme partition étendue.

Enfin, la structure logique de chaque partition dépend directement du système de fichiers qu'elle abrite.

6 **boot sector**: en anglais. Rappelez-vous du terme **boot** (**amorce** ou **démarrage**), il revient fréquemment.
7 **Master Boot Record**: à peu près *enregistrement principal pour le démarrage*, en français. On utilise plus volontiers le terme anglais ou l'abréviation **MBR**.

Le ***Tableau 3*** reprend l'**Exemple** de la ***Figure 8***, en montrant de quelle façon les partitions s'enchaînent les unes derrières les autres. Les références **/dev/sda**x sont données à titre d'exemple, et représentent les partitions d'un disque **SCSI** sur un système Linux.

Les disques **IDE** utilisent la nomenclature **hd** et les disques **SCSI** la nomenclature **sd**.

Partitions		Dispositif (*device*)		OS	Type	Contenu
/dev/sda						
Première	Primaire	/dev/sda1		*Win-7*	NTFS	
Deuxième	Étendu	/dev/sda2	/dev/sda5	*Win-7*	NTFS	*Table de partition logique*
						Première partition logique
			/dev/sda6	*Win-8.1*	NTFS	*Table de partition logique*
						Deuxième partition logique
			/dev/sda7	*Linux*	*SWAP*	*Table de partition logique*
						Troisième partition logique
			/dev/sda8	*Linux*	EXT2	*Table de partition logique*
						Quatrième partition logique
Troisième	Primaire	/dev/sda3		*Win-8.1*	NTFS	
Quatrième	Primaire	/dev/sda4		*Linux*	EXT4	

Tableau 3: Structure logique d'un disque dur SCSI.

Utilisation de la commande **fdisk** pour afficher les partitions d'un disque **SCSI** d'un système Linux.

```
[root@tchana ~]# fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Commande (m pour l'aide):
```

Affichage de l'aide.

```
Commande (m pour l'aide): m

Commande d'action
   a   bascule le fanion d'amorce
   b   éditer l'étiquette BSD du disque
   c   basculer le fanion de compatibilité DOS
   d   supprimer la partition
   l   lister les types de partitions connues
   m   afficher ce menu
   n   ajouter une nouvelle partition
   o   créer une nouvelle table vide de partitions DOS
   p   afficher la table de partitions
   q   quitter sans enregistrer les changements
   s   créer une nouvelle étiquette vide pour disque de type Sun
   t   modifier l'id de système de fichiers d'une partition
   u   modifier les unités d'affichage/saisie
   v   vérifier la table de partitions
   w   écrire la table sur le disque et quitter
   x   fonctions avancées (pour experts seulement)

Commande (m pour l'aide):
```


Affichage de la **Table des partitions** d'un disque **SCSI** d'un système Linux.

```
Commande (m pour l'aide):p

Disque /dev/sdb: 4294 Mo, 4294967296 octets
255 têtes, 63 secteurs/piste, 522 cylindres
Unités = cylindres de 16065 * 512 = 8225280 octets
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identifiant de disque : 0xe474cc36

Périphérique Amorce  Début        Fin      Blocs     Id  Système
/dev/sda1               1         65      522081    7  HPFS/NTFS
/dev/sda2              66        321     2056320    5  Etendue
/dev/sda3             322        386      522112+   7  HPFS/NTFS
/dev/sda4             387        451      522112+  83  Linux
/dev/sda5              66        130      522081    7  HPFS/NTFS
/dev/sda6             131        195      522081    7  HPFS/NTFS
/dev/sda7             196        260      522081   82  Linux swap / Solaris
/dev/sda8             261        321      489951   83  Linux

Commande (m pour l'aide):
```

5. Les mémoires

Le sujet de la mémoire dans un ordinateur est suffisamment vaste pour lui consacrer un ouvrage entier. Nous n'aborderons ici que des notions très générales afin de clarifier quelques termes courants.

5.1. La mémoire centrale ou RAM

La mémoire centrale ou **RAM**[8] de l'ordinateur est utilisée pour contenir le code exécutable et les données des programmes que vous utilisez, y compris le système lui-même. Sa caractéristique principale est que son contenu est irrémédiablement perdu lorsqu'on coupe l'alimentation électrique de l'ordinateur, d'où son nom courant de "**mémoire vive**".

A l'heure actuelle, le temps d'accès moyen à une information contenue dans la mémoire centrale est de l'ordre de quelques nanosecondes (*milliardième de seconde*). Une taille usuelle pour la mémoire centrale est de **4 Go**. Il est évident que plus cette taille est importante, plus le système peut l'utiliser pour stocker des informations auxquelles il accédera très rapidement par la suite.

5.2. La mémoire CMOS

Cette mémoire est utilisée pour conserver des paramètres généraux de la machine lorsque celle-ci est arrêtée, tels que la configuration du ou des disques durs, l'heure, la date, etc. Elle est entretenue par une pile branchée sur la carte mère et son contenu est perdu si on retire la pile ou si elle s'épuise. Mais inutile de paniquer, la durée de vie d'une telle pile est en général d'au moins quatre à cinq ans...

5.3. La mémoire vidéo ou VRAM

Cette mémoire est spécialisée pour l'affichage de l'écran. La quantité dont vous disposez détermine directement le nombre de couleurs et la finesse de graphisme que vous pouvez obtenir sur votre écran. Elle a généralement des temps d'accès très faibles, de l'ordre de quelques nanosecondes, sauf sur certains systèmes où elle n'est rien d'autre qu'une portion de la mémoire centrale réservée à cet usage (*ce que nous déconseillons pour des questions évidentes de performances générales*).

5.4. La mémoire cache

Il serait plus juste de parler des mémoires cache. Ces mémoires, qui se trouvent un peu partout, sont utilisées comme intermédiaires entre des composants relativement lents de la machine et d'autres plus rapides. Une de

8 **RAM**: **R**andom **A**ccess **M**emory, mémoire à accès libre en lecture et écriture.

ses utilisations est de stocker des informations fréquemment utilisées pour permettre des accès plus rapides; par exemple entre le disque dur (l*ent, temps d'accès de l'ordre de millièmes de seconde*) et la mémoire centrale (*environ un million de fois plus rapide!*).

Un exemple particulièrement important pour la puissance générale d'un système est la mémoire cache du processeur: il s'agit d'une mémoire ultra-rapide, bien plus que la mémoire centrale. Elle est utilisée pour stocker des informations venant de cette dernière fréquemment utilisées, telles que des données ou des portions de code exécutable.

5.5. La mémoire virtuelle

Imaginez la situation suivante: vous travaillez sur une machine dotée de 4Go de mémoire centrale, dont 3,5Go sont déjà occupés par divers programmes. Vous souhaitez lancer un logiciel qui a besoin d'au moins 1Go de mémoire centrale pour fonctionner. Vous allez donc vous trouver en "déficit" de 500Mo de mémoire.

Sur d'anciens systèmes tel que MS-DOS, cela se passe très simplement: le système refuse simplement de lancer le logiciel. Les systèmes modernes (*dont fait partie Linux*) sont plus subtils, et utilise le mécanisme de la mémoire virtuelle[9]. L'idée est d'utiliser une partie du disque dur comme s'il s'agissait de mémoire centrale, dont la taille paraît ainsi augmentée. Ce mécanisme vous permet d'utiliser plus de mémoire que votre machine n'en dispose réellement. Si cela est fort utile, surtout pour les machines disposant de peu de mémoire, gardez toutefois à l'esprit que le temps d'accès à une information sur le disque dur se mesure en millième de seconde tandis qu'il se mesure en milliardième de seconde pour la mémoire... Quand le système commence à utiliser ce mécanisme (*on dit qu'il "swap"*), attendez-vous donc à des performances fortement dégradées.

6. Autres matériels et périphériques

Un système d'exploitation digne de ce nom doit permettre à son propriétaire d'utiliser tous ses périphériques sans exception... du moins en théorie. En effet, la pratique est nettement moins drôle, surtout pour ceux qui osent sortir des sentiers battus en faisant confiance à Linux. Et là, les choses se gâtent.

- Il faut déjà oublier les "win-périphériques", en général des modems et des imprimantes, conçus pour n'être utilisés que dans un environnement déterminé.
- Il faut également tirer un trait sur le fameux pilote (*driver*) qui n'est fourni, en général, que pour le même environnement précité.
- Il faut faire confiance à la "communauté" Linux qui développe chaque jour de nouveaux pilotes pour faire fonctionner votre périphérique (*avant*-) dernier cri.

9 **virtuelle**: swap memory en anglais.

IV- Le système de fichiers de Linux en détail

1. Introduction

Nous détaillons ici quelques aspects importants du système de fichiers de Linux. Il n'est pas utile de parfaitement maîtriser toutes les notions abordées ici mais, au moins nous vous recommandons d'en avoir une connaissance générale.

2. Accès aux périphériques, au matériel

Comme nous l'avions évoqué au début de ce document, les systèmes Unix - et donc Linux - accèdent aux éléments matériels de la machine par l'intermédiaire de fichiers spéciaux.

Sous Linux, ces fichiers sont tous regroupés dans le répertoire **/dev** (*comme device, périphérique en français*). Le principe de fonctionnement de ces fichiers est un peu particulier et leurs noms sont (*plus ou moins*) normalisés. Un fichier spécial du répertoire **/dev** est caractérisé par deux nombres, les numéros majeur et mineur.

2.1. Disque IDE

Considérons l'affichage suivant:

```
[root@tchana ~]# ls -l /dev/hda

brw-rw-- 1  root  disk  3,   0  6 janv. 10:01 /dev/hda
[root@tchana ~]#
```

```
[root@tchana ~]# ls -l /dev/ttyS1

crw-r-r- 1  root  root  4,  65  6 janv. 10:02 /dev/ttyS1
[root@tchana ~]#
```

Les numéros majeurs et mineurs apparaissent juste avant la date. Décrivons-les.

Le numéro majeur correspond à un périphérique donné: par exemple, **03** correspond au premier port **IDE** sur la carte mère, et les disques associés sont désignés par des fichiers dont le nom commence par **hda** pour le disque maître, **hdb** pour le disque esclave. Ainsi, le fichier **/dev/hda** est un accès direct au premier disque dur, pris dans sa globalité, en tant que suite ininterrompue d'octets pratiquement non structurée. Inutile de préciser qu'il vaut mieux éviter d'écrire dans ce fichier! Le début est notamment la **Table des Partitions**, à manier avec délicatesse... Voir le paragraphe ***Partition des disques durs*** à la page ***32*** pour plus de détails concernant les partitions des disques.

Le numéro mineur correspond à une subdivision du périphérique. Dans le cas du port **IDE**, les mineurs à partir de **1** numérotent les partitions du premier disque (*le disque maître*), les mineurs à partir de **65** numérotent les partitions du second disque (*le disque esclave*). Par exemple, **/dev/hda5** (*dont on note les caractéristiques par **03:05***) désigne la première **partition logique** du disque dur maître sur le premier port **IDE**, tandis que **/dev/hdb2** (*dont on note les caractéristiques par **03:66***) désigne la deuxième partition primaire du disque dur esclave sur le premier port **IDE**.

2.2. Disque SCSI

```
[root@tchana ~]# ls -l /dev/sd*

brw-rw---- 1 root disk 8, 0 Dec 17 12:05 /dev/sda
brw-rw---- 1 root disk 8, 1 Dec 17 12:05 /dev/sda1
brw-rw---- 1 root disk 8, 2 Dec 17 12:05 /dev/sda2
[root@tchana ~]#
```

Dans cet exemple, nous avons quatre disques **SCSI**. Chaque disque a deux partitions.

Le numéro majeur **8** correspond au port **SCSI** sur la carte mère, et les disques associés sont désignés par des fichiers dont le nom commence par **sda** pour le premier disque, **sdb** pour le deuxième disque, etc. Ainsi, le fichier **/dev/sda** est un accès direct au premier disque dur, pris dans sa globalité, en tant que suite ininterrompue d'octets pratiquement non structurée.

2.3. Disquette

Sauf situation assez exceptionnelle, vous n'aurez que très rarement à utiliser directement ces fichiers spéciaux. Un exemple d'une telle situation est lorsque le besoin de prendre une image d'une disquette se présente, par exemple pour la dupliquer.

Le fichier spécial correspondant au premier lecteur de disquette est **/dev/fd0** (*02:00*). Pour dupliquer une disquette en conservant très précisément sa structure (*notamment s'il s'agit d'une disquette de démarrage*), la première étape consiste à prendre une image de la disquette:

```
[root@tchana ~]# cp /dev/fd0 disque.img

[root@tchana ~]#
```

Cette commande (*notez l'utilisation de la commande usuelle **cp***) va lire la disquette octet par octet (*ou plutôt secteur par secteur*), sans s'occuper de sa structure et encore moins du système de fichier qu'elle contient (*qu'il soit de type Linux, MS-DOS, ou autre*). Le résultat est placé dans le fichier **disque.img** dans le répertoire courant. Ce fichier est ce qu'on appelle une image de la disquette (*d'où son extension **.img***).

La seconde étape (*après avoir changé la disquette dans le lecteur!*) est exactement symétrique de la première:

```
[root@tchana ~]# cp disque.img /dev/fd0

[root@tchana ~]#
```

... va écrire le fichier image sur la disquette dans le lecteur. Vous aurez alors une copie absolument exacte de la disquette originale, au bit près.

Ceci peut évidemment se généraliser aux disques durs ou **CD-ROMs**.

Mais attention! Si vous avez une partition désignée par exemple par **/dev/hda7**, d'une taille de 900 Mo, la commande:

```
[root@tchana ~]# cp /dev/hda7 partition-hda7.img

[root@tchana ~]#
```

... va vous créer un fichier de 900 Mo dans le répertoire courant! Donc prenez garde à avoir l'espace nécessaire...

2.4. Character vs block

Enfin et pour finir, Linux effectue une distinction entre les périphériques d'entrée-sortie par caractères (*character devices*) et les périphériques d'entrée-sortie par blocs (*block devices*). La différence apparaît dans les exemples plus haut sur le premier caractère de la ligne résultant du **ls**: '**c**' pour caractères, '**b**' pour blocs. Les pé-

riphériques d'entrée-sortie par caractères ont comme caractéristique de transmettre et recevoir les informations octet par octet. C'est le cas des ports séries ou parallèles, des modems, etc. Par contre, les périphériques d'entrée-sortie par blocs transmettent ou reçoivent les informations sous forme de paquets d'octets, d'une taille fixe: c'est le cas des supports de mémoire de masse (*disquettes, disques durs...*). Ainsi, un même couple **majeur/mineur** peut désigner deux périphériques différents, selon que l'on considère un périphérique par **caractères** ou un périphérique par **blocs**.

A titre indicatif, voici quelques fichiers avec les périphériques associés dans le ***Tableau 4***. La colonne **B/C** indique s'il s'agit d'un périphérique de caractères ou un périphérique par **blocks**.

Fichier	Majeur	Mineur	B/C	Périphérique
/dev/mem	1	1	c	accès direct à la mémoire centrale
/dev/fd0	2	0	b	premier lecteur de disquettes
/dev/hda	3	0	b	disque maître sur le premier port IDE
/dev/hda2	3	2	b	seconde partition primaire sur ce disque
/dev/hdb	3	64	b	disque esclave sur le premier port IDE
/dev/hdb5	3	69	b	première partition logique sur ce disque
/dev/tty1	4	1	c	première console virtuelle
/dev/lp0	6	2	c	troisième port parallèle (*imprimante*)
/dev/sda	**8**	**0**	**b**	premier disque dur SCSI
/dev/sda2	**8**	**2**	**b**	deuxième partition sur ce disque
/dev/sdb	**8**	**16**	**b**	deuxième disque dur SCSI
/dev/sdb1	**8**	**17**	**b**	deuxième partition sur ce disque
/dev/psaux	10	1	c	port PS/2 (*souris*)
/dev/kdb	11	0	c	accès direct au clavier
/dev/scd0	11	0	b	premier CD-ROM SCSI
/dev/sequencer	14	1	c	séquenceur de la carte son
/dev/hdc	22	0	b	disque maître sur le second port IDE
/dev/video0	81	0	c	Acquisition vidéo

Tableau 4: Quelques fichiers spéciaux et les périphériques associés.

et ainsi de suite... la liste complète occuperait plusieurs pages!

Pour la liste complète des numéros **majeur** et **mineur**, voir:
http://www.linux-france.org/article/kafkafr/node19.html

2.5. L'arborescence unique et les systèmes de fichiers

2.5.1. notion de montage

Affirmons-le encore une fois, sous le système Unix, et donc sous Linux, tout est fichier. Notamment, toutes les informations auxquelles accède le système sont incluses dans une arborescence unique; qu'elles soient sur le même support physique que le système ou sur des supports différents (*autres partitions, disques,* ***CD-ROMs,*** *etc.*)

D'où la notion de système de fichiers: un système de fichier, en gros, désigne une partition sur un support physique quelconque. Dans le cas d'une disquette ou d'un **CD-ROM**, désigne en général tout le disque. Cette notion peut être rapprochée de celle évoquée en début de document, au paragraphe ***Système de fichiers*** à la page ***9***; une partition donnée applique en général un seul système de fichier (*au sens du paragraphe* ***Système de fichiers****, donc* ***FAT16****,* ***NTFS****,* ***EXT3FS****, ...*), donc la confusion des deux notions est possible - et très souvent effectuée.

Lorsque l'on désire accéder aux informations d'un support donné (*par exemple, un* ***CD-ROM***), il est nécessaire, en quelque sorte, "d'accrocher" le système de fichier qu'il contient à un point de l'arborescence, c'est-à-dire un répertoire (*de préférence prévu à cet effet*). Cette opération s'appelle le montage[10] du support en question. Une fois cette opération effectuée, les fichiers et répertoires du support apparaissent comme se trouvant dans le répertoire sur lequel le support a été monté.

Par exemple, vous possédez un **CD-ROM** contenant à sa racine le fichier **liste.txt**, ainsi que le répertoire **Images**. Si vous montez le **CD-ROM** sous le répertoire **/mnt**, l'accès au fichier se fera par **/mnt/liste.txt**, l'accès au répertoire par **/mnt/Images**, l'accès aux fichiers du répertoire **Images** par **/mnt/Images/nom_de_fichier**, et ainsi de suite.

Notez bien que si le répertoire **/mnt** contenait des fichiers avant le montage, ceux-ci deviennent absolument inaccessibles une fois le montage effectué. Prenons un exemple; supposons un répertoire **/mnt**, qui contient des fichiers:

```
[root@tchana ~]# ls /mnt

cdrom dos1 dos2 eit floppy iomega nfs
[root@tchana ~]#
```

Montons un **CD-ROM** sur ce répertoire (*la commande utilisée est* ***mount****, elle sera détaillée par la suite*), et examinons le contenu du répertoire:

```
[root@tchana ~]# mount /dev/hdd /mnt

mount: block device /dev/hdd is write-protected, mounting read-only
[root@tchana ~]#
```

```
[root@tchana ~]# ls /mnt

3DFx           GTK1.2            Koffice       Sound
Alien          Gnome1.0_Sources  Lisezmoi.txt  Themes
ContribSuite   Java1.2_pre1      Noyaux        docs
Dev_Graphique  Jeux              Revue         kde
EnVrac         KDE1.1            SciTech
[root@tchana ~]#
```

On constate que les fichiers qui se trouvaient dans **/mnt** ont comme "disparus".

10 **montage**: action de monter: to mount, en anglais.

Mais que faire lorsque l'on souhaite changer le **CD-ROM**? Il existe l'opération inverse du montage, il s'agit - vous l'aurez deviné du démontage[11] des systèmes de fichiers (*avec **umount**, détaillée plus loin*). Si le point de montage contenait des fichiers avant le montage, ceux-ci redeviennent visibles:

```
[root@tchana ~]# umount /mnt

[root@tchana ~]#
```

```
[root@tchana ~]# ls /mnt

cdrom dos1 dos2 eit floppy iomega nfs
[root@tchana ~]#
```

Prenons un exemple réel. Il est assez commun, sur un système Linux, de répartir les différents éléments du système sur différents systèmes de fichiers, pas nécessairement sur le même disque. À la ***Figure 10***, les éléments sur un même système de fichiers sont entourés de pointillés, le tout constituant un système Linux complet (*l'arborescence est ici simplifiée*).

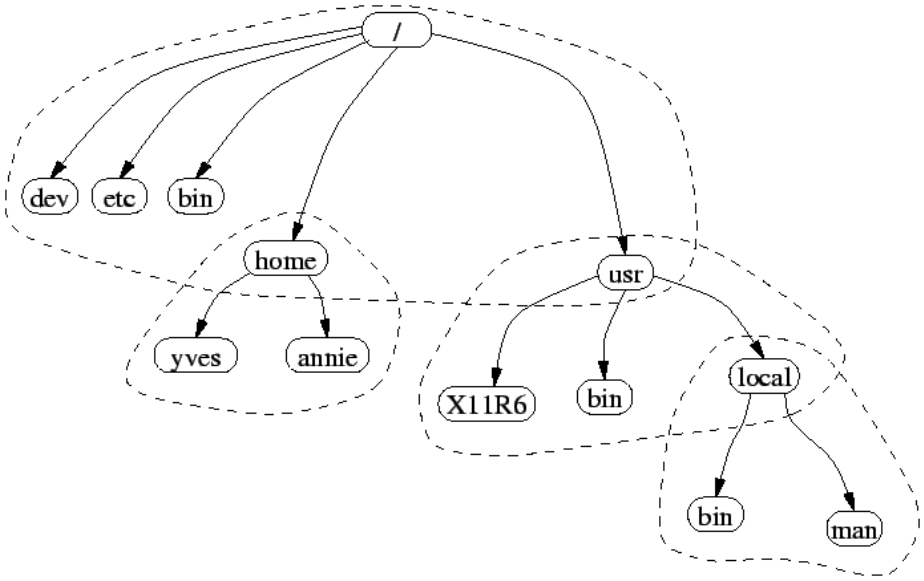


Figure 10: Différents systèmes de fichiers pour un système Linux.

Pour finir, disons simplement qu'il est usuel de réserver plusieurs sous-répertoires dans le répertoire **/mnt** pour les montages temporaires, tels que disquettes et **CD-ROM**s, par exemple, **/mnt/cdrom**, **/mnt/floppy**, etc.

Deux commandes sont dédiées aux opérations de montage et démontage, les commandes **mount** et **umount**, exposées au paragraphe ***Montage et démontage de systèmes de fichiers: mount et umount*** à la page ***60***.

11 **démontage**: démonter: to unmount, en anglais

3. Mécanisme des liens

Les principes de fonctionnement du système de fichiers de Linux comporte la notion de **liens**. Pour être simple, un lien[12] est un point d'accès à un fichier. Ce point d'accès prenant la forme d'un nom dans un répertoire - la notion de fichier ne représente qu'une suite de caractères quelque part sur un disque. Quelle différence alors, entre le nom de fichier tel qu'exposé au début de ce document et le lien? En fait, le nom de fichier est un élément du lien qui comporte de nombreuses informations concernant le fichier (*droits d'accès, taille, date de création...*). De plus, nous avions évoqué une sorte d'équivalence entre le nom de fichier et le fichier lui-même. Mais un même fichier peut être référencé par plusieurs liens de noms différents s'ils sont dans le même répertoire, éventuellement de mêmes noms s'ils sont dans des répertoires différents.

On distingue deux types de liens: les liens physiques et les liens symboliques[13].

Un **lien physique** est un accès direct à la suite de caractères sur disque qu'est le fichier. Accéder au lien, c'est accéder directement au fichier. En pratique, lorsque vous effacez un fichier avec **rm**, vous indiquez un nom de fichier, donc un lien. Mais l'espace occupé par le fichier n'est effectivement libéré que lorsque le dernier lien physique qui y fait référence est effacé: **rm** ne supprime donc pas le fichier, seulement un lien qui lui est attaché. Ceci est une façon de protéger des fichiers contre l'effacement, mais rend l'utilisation des liens physiques délicates: il est facile d'engendrer un véritable capharnaüm dans l'arborescence...

Un **lien symbolique**, à contrario, est un accès indirect au fichier; on peut plutôt le voir comme une référence à un autre lien. Comme un lien secondaire sur le fichier correspondant. La grosse différence avec le lien physique est qu'il est possible d'effacer le fichier (*c'est-à-dire effacer tous ses liens physiques*) sans toucher aux liens symboliques (*qui deviennent alors en quelque sorte orphelins; on parle de liens brisés*). D'un maniement plus souple que les liens physiques, les liens symboliques peuvent également amener à une forêt inextricable de références, n'en abusez donc pas.

Une commande est dédiée à la manipulation des liens, la commande **ln**, présentée au paragraphe ***Manipulation des liens: ln*** à la page ***60***.

4. Utilisateurs: droits et devoirs

Linux est un système multi-utilisateur. C'est-à-dire que tout est prévu dans le système pour que plusieurs personnes puissent l'utiliser, même simultanément (*dans le cas de configurations en réseau*), sans se gêner les uns les autres.

Chaque utilisateur est identifié par un nom, et doit fournir un **mot de passe**[14] pour pouvoir utiliser le système. La procédure au cours de laquelle un utilisateur donne son nom et son mot de passe est appelée "**procédure de login**", ou simplement "**login**". Ce mot de passe est strictement personnel: théoriquement, vous ne devez communiquer le vôtre à personne, encore que cela ne prête pas beaucoup à conséquence sur une machine individuelle (*sauf si vous possédez des fichiers confidentiels...*)

De même, chaque utilisateur se voit attribué une certaine zone dans l'arborescence des répertoires (*typiquement, un sous-répertoire dont le nom est celui de l'utilisateur dans le répertoire **/home***), que l'on appelle parfois le "répertoire personnel[15]"de l'utilisateur. L'utilisateur possède un pouvoir quasiment absolu de création, modification et destruction sur les fichiers de son répertoire personnel (*fichiers dont il est le propriétaire*), mais normalement il ne peut pas intervenir sur les répertoires des autres utilisateurs et encore moins sur les éléments constitutifs du système. Ceci confère à Linux un haut degré de sécurité vis-à-vis des diverses manipulations et autres fausses manoeuvres de ceux qui l'utilisent.

Afin de faciliter le travail coopératif, les utilisateurs peuvent êtres regroupés par **groupes**. Il est ainsi possible d'autoriser tout un groupe à accéder à certains fichiers, mais pas les autres utilisateurs.

12 **liens**: link en anglais.
13 **symboliques**: respectivement, hard links et symbolic links en anglais.
14 **mot de passe**: password en anglais.
15 **répertoire personnel**: home directory en anglais.

4.1. Le super-utilisateur

Comme indiqué plus haut, normalement les utilisateurs ne peuvent accéder aux éléments constitutifs du système. Comment alors, accéder à ceux-ci quand le besoin s'en fait sentir, comme lors de l'installation d'un nouveau logiciel?

Pour les opérations nécessitant une modification particulière du système, il existe un utilisateur particulier, privilégié, le **super-utilisateur**[16], dont le nom est **root**. En fait, l'utilisateur root possède un pouvoir total sur l'intégralité du système, y compris celui de le détruire en quelques secondes. Il peut consulter, modifier et détruire n'importe quel fichier du système et on peut estimer qu'en pratique, root est implicitement propriétaire de tous les fichiers du système.

Si vous êtes seul à utiliser Linux sur votre machine personnelle, l'utilisateur root, c'est vous. Donc lorsque vous utilisez votre machine en tant que root, il convient de prendre un luxe de précautions et de bien savoir ce que vous faites; la précipitation est le chemin le plus sûr vers le massacre de votre système et la perte de vos données. C'est pourquoi il est vivement recommandé de créer au moins un autre utilisateur pour l'utilisation quotidienne du système. Une erreur de manipulation n'aura alors que des conséquences limitées sur l'intégrité du système.

4.2. Droits des fichiers et des répertoires

Pour réaliser la protection des fichiers évoquée plus haut, il existe un mécanisme de droits et de propriété attribués à chacun des fichiers du système. Tout fichier est la propriété d'un utilisateur et les droits s'articulent autour de cette notion.

4.3. Droits généraux: lecture, écriture, exécution

Linux (*comme Unix*) reconnaît trois droits fondamentaux.

- le droit de lecture, symbolisé par la lettre '**r**' (*pour read*), indique si l'on est autorisé à consulter le fichier, c'est-à-dire, par exemple, à afficher son contenu par une commande telle que **more**.
- le droit de écriture, symbolisé par la lettre '**w**' (*pour write*), indique si l'on est autorisé à modifier le contenu du fichier, par exemple à l'aide d'un programme tel que **vi**.
- le droit de exécution, symbolisé par la lettre '**x**' (*pour execute*), indique si l'on peut exécuter le fichier comme s'il s'agissait d'une commande. Ceci n'a évidemment de sens que pour les fichiers contenant des instructions compréhensibles par le système comme c'est le cas de fichiers tels que: **/bin/more**, **/sbin/fdisk**...

Les droits individuels d'un fichier peuvent être modifiés à l'aide de la commande **chmod**, exposée au paragraphe ***Modifier les droits sur un fichier ou un répertoire: chmod*** à la page ***59***.

4.4. Regroupement des droits

Les droits fondamentaux qui viennent d'être exposés sont, pour chaque fichier, présents dans trois groupes de droits distincts.

- le premier groupe, symbolisé par la lettre '**u**' (*pour user*), précise les droits pour le propriétaire du fichier.
- le deuxième groupe, symbolisé par la lettre '**g**' (*pour group*), précise les droits pour les utilisateurs du groupe auquel appartient le propriétaire du fichier.
- le troisième groupe, symbolisé par la lettre '**o**' (*pour other*), précise les droits pour, disons, le reste du monde, c'est-à-dire n'importe quel utilisateur (*autre que le propriétaire du fichier et les utilisateurs de son groupe*).

Par exemple, la commande **more** (*voir le paragraphe* ***Affichage de fichiers (texte)*** *à la page* ***26***), normalement accessible à tout le monde, possède le droit en exécution du groupe **other** activé (*de même que les droits en exécution des deux autres groupes*). Par contre, la commande **dd** (*voir le paragraphe* ***Entrées/sorties directes***

16 **super-utilisateur**: super-user en anglais.

de données: dd *à la page* *62*) ne devrait avoir que le droit en exécution du propriétaire (en occurrence, **root**) d'activé.

Le propriétaire d'un fichier peut être modifié avec la commande **chown**.

4.5. Cas particulier des répertoires

Lorsque le fichier considéré est un répertoire, les droits prennent une signification un peu différente.

- le droit en lecture indique que l'on peut obtenir la liste des fichiers du répertoire, par la commande **ls** par exemple.
- le droit en écriture indique que l'on peut créer ou supprimer des fichiers du répertoire (*la modification ou la consultation de ceux-ci dépend des droits de chaque fichier*).
- le droit en exécution indique si l'on peut "transiter" par le répertoire, c'est-à-dire le faire figurer dans une succession de répertoires dans un chemin, ou bien en faire le répertoire courant. Ceci est totalement indépendant des deux droits précédents.

5. Fichiers impliqués dans la gestion des utilisateurs

5.1. Le fichier des utilisateurs et de leur mot de passe: /etc/passwd

Ce fichier contient la liste des utilisateurs qui peuvent utiliser le système. **Attention!** Ne le supprimez pas. Si vous le supprimez, il vous sera impossible de vous connecter au système, même en tant que **root**.

V- Les variables d'environnement

1. Introduction

Il arrive parfois que certaines informations doivent être mise à disposition d'un grand nombre de programmes. Par exemple, nous avons vu que lorsque vous exécutez une commande (*telle qu'un simple "**ls**"*), c'est en réalité un programme qui est exécuté. Or ce programme est contenu dans un fichier qui doit être lu pour que le système puisse exécuter le programme. Mais comment le système fait-il pour trouver le fichier en question? L'information indiquant à quels endroits chercher les fichiers contenant les programmes des diverses commandes doit donc être aisément accessible et, comme certaines commandes en appellent d'autres, cette information doit également être lisible par tout le monde.

Ce genre d'information largement diffusée dans le système est contenu dans ce que l'on appelle des variables d'environnement. Une variable est une zone de la mémoire, identifiée par une étiquette (*un nom*) et contenant quelque chose (*un nombre, un mot...*) Ces variables sont fixées, certaines au démarrage du système et d'autres lors de la connexion d'un utilisateur. Examinons l'une d'entre elle.

2. PATH

Exécutez la commande suivante:

```
[root@tchana ~]# echo $PATH

/sbin/e-smith:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/puppetlabs/bin:/root/bin
[root@tchana ~]#
```

La commande **echo** permet d'afficher quelque chose. Dans la commande précédente, nous lui demandons d'afficher le contenu d'une variable dont le nom est **PATH**. Notez l'utilisation du symbole dollar ('**$**') devant le nom de la variable, nécessaire lorsque l'on souhaite accéder au contenu de la variable.

C'est manifestement là une suite de répertoires, donnés par leur chemin absolu et séparés par le caractère deux-points ('**:**'). Nous avons là précisément la réponse au problème posé au début de ce paragraphe. La variable d'environnement **PATH** contient une liste de répertoires dans lesquels chercher les fichiers contenant les programmes des commandes que l'on souhaite exécuter.

Supposons maintenant que vous souhaitiez modifier cette variable pour lui rajouter le répertoire **/toto**. Si vous utilisez le **shell Bash**, la commande à utiliser est:

```
[root@tchana ~]# export PATH="$PATH:/toto"

[root@tchana ~]#
```

Notez les deux modes d'emploi du nom de la variable. D'abord sans le dollar à gauche du signe égal, parce que l'on fixe sa valeur puis, avec le dollar "$", parce que l'on lit sa valeur. Vérifions ce que nous venons de faire.

```
[root@tchana ~]# echo $PATH

/sbin/e-smith:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/puppetlabs/bin:/root/bin:/toto
[root@tchana ~]#
```


Ce qui est bien le résultat escompté.

Si par contre vous utilisez le **C-shell**, la commande est:

```
setenv PATH "${PATH}:/toto"
```

Le résultat est le même, la variable **PATH** est modifiée.

Il existe un grand nombre de variables d'environnement, chacune destinée à un usage particulier. Chaque shell actif possède son propre ensemble de variables d'environnement. Si deux utilisateurs sont connectés simultanément sur la même machine, on a deux shells actifs. Chacun des shells possède ses propres variables d'environnement qui ne sont pas forcément les mêmes. Mieux, une variable donnée peut avoir deux valeurs différentes selon l'utilisateur.

Pour obtenir l'ensemble des variables d'environnement définies lorsque vous êtes connectés avec leurs valeurs, utilisez la commande "**env**".

```
[root@tchana ~]# env

env
XDG_SESSION_ID=5
HOSTNAME=pc-00075.micronator.org
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=192.168.1.81 61939 22
SSH_TTY=/dev/pts/0
USER=root
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=
01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=
01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.
tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=
01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar
=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*
.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35
:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=
01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*
.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;3
5:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;
35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=
01;36:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mpc=01;36:*.ogg=01;36:*.
ra=01;36:*.wav=01;36:*.axa=01;36:*.oga=01;36:*.spx=01;36:*.xspf=01;36:
MAIL=/var/spool/mail/root
PATH=/sbin/e-smith:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/puppetlabs/bin:/root/bin
PWD=/root
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
SHLVL=1
HOME=/root
LOGNAME=root
SSH_CONNECTION=192.168.1.81 61939 192.168.1.75 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/0
HISTTIMEFORMAT=%Y-%m-%d %T
_=/usr/bin/env
[root@tchana ~]#
```


VI- Fichiers particuliers

1. Les runlevels

Les **niveaux d'exécution**[17], au nombre de **7** dans NethServer, indiquent au système de quelle façon il doit s'initialiser ou, plus précisément, dans quel état de configuration il doit se trouver lors de son fonctionnement. On peut passer d'un niveau à l'autre avec la commande **init**.

Pour passer au niveau d'exécution numéro **5**.

```
init 5
```

Seuls les deux niveaux extrêmes sont normalisés dans Linux:

- le niveau **0** (*halt*) est celui dans lequel doit passer le système lorsqu'on désire arrêter la machine afin de le faire "proprement", notamment en vérifiant que toutes les informations qui doivent être écrites sur le disque, l'ont effectivement été.
- le niveau **6** (*reboot*) est identique au niveau **0**, mais en plus il provoque un **redémarrage à chaud** de l'ordinateur (*c'est un équivalent de la combinaison de touches **[Ctrl]+[Alt]+[Suppr]***).

Mis à part les deux précédents, chaque distribution Linux dispose des autres niveaux comme elle l'entend. Nous prendrons ici l'exemple d'une distribution **RedHat**, mais ce qui suit est aisément transposable aux autres distributions.

Parmi les niveaux d'exécution, l'un d'eux est défini comme niveau par défaut, C'est le niveau qui est sélectionné lors du démarrage du système. Le lecteur astucieux aura compris qu'il n'est pas conseillé de fixer le niveau par défaut à **0** ou à **6**...

Pour chaque niveau d'exécution, il existe un ensemble de scripts (*des fichiers textes exécutables, en réalité des programmes en langage Shell*) qui réalisent l'initialisation des différents services du système: configuration du clavier, du serveur mail, de la souris... Le niveau d'exécution par défaut permet de déterminer quels seront les scripts à exécuter et ainsi de spécialiser le comportement du système selon ce que vous souhaitez faire.

Par exemple, sur une distribution **RedHat**:

- le niveau **1** est un mode mono-utilisateur; seul **root** peut accéder à la console. Ce mode n'est normalement utilisé que pour des opérations de maintenance sensibles qui nécessitent un fonctionnement minimal du système pour éviter les interférences dues à d'autres utilisateurs. Pratiquement aucun script de démarrage n'est exécuté.
- le niveau **2** est un mode multi-utilisateur, sans support réseau. On retrouve les six consoles textes virtuelles usuelles (*voir le paragraphe **Notion de console** à la page **14***). Les scripts spécifiques à la gestion du réseau ne sont pas exécutés.
- le niveau **3** est identique au niveau **2**, mais avec le support réseau. C'est le mode de fonctionnement usuel.
- le niveau **4** est par défaut identique au niveau **3**, mais vous pouvez librement l'utiliser pour vos propres besoins pour des configurations personnalisées.

17 **runlevels**: niveaux d'exécution, en français.

- le niveau **5**, enfin, est identique au niveau **3**, mais provoque un démarrage en mode graphique sous la couche **X-Window**.

Le choix du niveau par défaut et les actions associées à chaque niveau sont inscrits dans un seul fichier qui est décrit ci-dessous.

2. Le fichier /etc/inittab

Avec **CentOS 7** *(NethServer)* et **RHEL 7**, le processus **systemd** remplace le processus **init** pour le démarrage des services à l'amorçage et pour la modification des niveaux d'exécution. Il utilise des **targets** au lieu de niveaux d'exécution et s'appuie sur la commande **systemctl** pour changer le niveau d'exécution ou le **target**. Le **systemd** fournit beaucoup plus de contrôle que le processus **init** tout en prenant en charge les scripts init existants.

Notez que dans **RHEL 7** et **Centos-7** *(NethServer-7)*, les modifications du fichier **/etc/inittab** ne prendront pas effet.

Pour les version antérieures à **Centos-7** *(NethServer-7)*, le fichier **/etc/inittab** est un simple fichier texte donc, modifiable avec n'importe quel éditeur de texte. Toutefois, si tout le monde peut le consulter, il faut être **root** pour le modifier.

Ce fichier est une suite de lignes ayant toutes la même forme générale et composées de quatre éléments (*on dit "champs*[18]*"*):

```
<id>:<niveau(x)>:<action>:<programme>
```

id

- est une suite, de un à quatre caractères, utilisée pour identifier la ligne. On ne doit pas retrouver deux fois la même suite sur deux lignes différentes.

niveau(x)

- indique quels sont les niveaux concernés par la ligne. Par exemple, '**124**' indique que la ligne ne concerne que les niveaux **1**, **2** et **4**.

action

- précise de quelle manière la ligne doit être interprétée, c'est-à-dire dans quel cadre elle s'inscrit ou de quelle façon traiter les autres champs.

programme

- est le programme à exécuter, dans le cadre précisé précédemment, pour les niveaux donnés.

Le plus simple, pour bien comprendre ces différentes notions, est d'examiner ensemble un exemple réel. Le fichier suivant est extrait d'un système **RedHat** Linux, version **5.2**:

```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
```

18 **champs**: fields en anglais.


```
#
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:12345:respawn:/sbin/mingetty -noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Avec la sortie de RedHat Enterprise Linux (**RHEL**) 6, RedHat utilisera le nouveau service de démarrage **Upstart**, en remplacement de l'ancien init.

La première chose, qui paraît manifeste, est que les lignes commençant par un dièse ('#') sont des **commentaires**. Leur présence est purement informative et ces lignes ne sont absolument pas prises en compte par le système. Ne négligez pas ces lignes, elles apportent beaucoup à la clarté des fichiers de configuration et autres scripts; il n'y en a jamais trop.

Considérons la première ligne "significative".

```
id:5:initdefault:
```

- Cette ligne précise quel niveau doit être choisi pour démarrer le système (*ici, le niveau 5*). C'est le sens de l'action '**initdefault**'. Ici, pas de programme associé, cela ne présente pas d'intérêt. Si vous supprimez cette ligne (*ou mieux, vous placez un dièse au début*), durant le démarrage le système demandera dans quel niveau il doit se placer pour terminer son initialisation. Cela peut être utile pour tester différentes configurations. Consultez la documentation de votre distribution pour connaître l'utilisation des niveaux de **1** à **6**, si ce n'est pas explicité comme ici dans un commentaire.

La ligne suivante.

```
si::sysinit:/etc/rc.d/rc.sysinit
```


- Indique un programme à exécuter dans les premiers temps du démarrage (*action **'sysinit'***), juste après que le matériel physique ait été reconnu. À ce stade, la notion de niveaux n'existe pas encore réellement, c'est pourquoi aucun n'est présent.

Viennent ensuite un certain nombre de lignes utilisées pour effectuer des actions spécifiques à chaque niveau. L'action '**wait**' indique que le processus d'initialisation ne se poursuivra pas tant que le programme spécifié ne sera pas terminé. Dans le cas présent c'est le même programme qui est appelé pour chaque niveau, mais avec un paramètre différent.

Le programme mentionné dans la ligne suivante, associé à l'action '**once**', sera exécuté dès l'entrée dans un nouveau niveau. Le vide de la liste des niveaux est ici interprété comme "**tous les niveaux**".

La ligne qui suit réalise l'interception de la séquence de touches **[Ctrl]+[Alt]+[Suppr]** qui, normalement, provoque un redémarrage de la machine. C'est effectivement ce qui se passe, mais de manière contrôlée par le système, donc "proprement".

Les deux lignes suivantes ne sont utiles que si vous disposez d'un **onduleur**, et qu'il est correctement configuré. La première ligne provoque un arrêt du système après un certain délai, en cas de coupure de courant. La seconde ligne arrête la procédure d'arrêt (*si cela est possible*) dans l'éventualité où le courant revienne.

Les six lignes suivantes effectuent la mise en place des consoles textes virtuelles (*voir le paragraphe **Notion de console** à la page **14** au sujet des consoles virtuelles*). En fait, le programme exécuté est celui qui demande le nom et le mot de passe de l'utilisateur pour contrôler l'accès au système. Comme à priori une telle procédure peut survenir plusieurs fois sur une même console (*plusieurs utilisateurs qui se connectent et se déconnectent les uns après les autres*), l'action '**respawn**' indique que le programme doit être relancé lorsque son exécution est terminée.

Notez que le niveau **1**, que nous avions décrit comme un mode mono-utilisateur, ne se voit attribué qu'une seule console virtuelle (*qui devient alors réelle...*)

Enfin, la dernière ligne est utilisée pour un démarrage en mode graphique.

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Pour plus de détails concernant ce fichier, consultez la page **man** correspondante: **man 8 inittab**.

3. Les scripts de démarrage

On désigne par ce terme, un ensemble de fichiers textes qui sont en fait des programmes en langage **shell**. Le rôle de ces programmes est de mettre en place la configuration générale de la machine lors du démarrage.

Par "mettre en place la configuration", on entend plusieurs choses:

- configuration du clavier, des polices d'écran, de la souris, de toutes ces petites choses qui nous simplifient la vie;
- démarrage de différents services (*appelés aussi "**daemons**[19]"*), c'est-à-dire les fonctionnalités que le système devra assurer durant son fonctionnement. ¨Par exemple: le serveur Internet, le courrier électronique, l'imprimante, les répertoires partagés...

Ces scripts se trouvent centralisés en un seul endroit, **/etc/rc.d**. Mais à partir de ce point, il n'existe pour l'instant pas de standard entre les différentes distributions (*bien qu'une normalisation soit en cours*). Encore une fois, pour conserver la cohérence avec ce qui précède, nous nous référerons à une distribution **RedHat**.

Cette distribution respecte une norme Unix, la norme dite **System V**. Disons tout de même ici que cette norme tant à se généraliser.

Le répertoire **/etc/rc.d** contient quelques fichiers et (*au moins*) huit répertoires. Décrivons, dans un premier temps, les fichiers (*qui sont en fait des programmes exécutables en langage **shell***):

19 De **daemon** (**D**isk **a**nd **E**xecution **Mon**itor) en anglais. En fait, services et démons sont deux choses différentes: le démon est le processus mémoire qui, par son exécution, rend un certain service (comme la gestion du courrier électronique). Mais la confusion est pour l'instant sans grande conséquence.

- rc
- rc.local
- rc.news
- rc.sysinit

Le sous-répertoire important est **init.d**: il contient tous les scripts de configuration des différents services qui doivent être lancés. En général, le nom du fichier exécutable correspondant à un certain service est assez explicite: par exemple, le fichier **httpd** lance (*ou arrête, cela dépend des paramètres*) le service qui permet à votre machine Linux de se comporter comme un serveur de pages **HTML** sur le **Word Wide Web**.

Viennent ensuite un certain nombre de sous-répertoires, dont le nom est de la forme **rcn.d**, le **n** correspond à un niveau d'exécution. Ainsi, si vous démarrez en niveau **5**, c'est le répertoire **rc5.d** qui va déterminer quels scripts sont lancés.

Chacun de ces sous-répertoires contient un ensemble de liens symboliques (*voir le paragraphe **Mécanisme des liens** à la page 42 pour plus d'information sur les liens*) vers certains des fichiers du répertoire **init.d**. Le nom des liens respecte un certain format:

- la première lettre doit être '**S**[20]' ou '**K**[21]', selon que le script doit être lancé à l'entrée ou à la sortie du niveau d'exécution;
- suivent deux chiffres, qui sont utilisés pour déterminer un certain ordre dans le lancement des scripts. Inutile de lancer la gestion du courrier électronique si le réseau n'est pas en place...
- enfin, plusieurs caractères qui sont en général le nom du fichier dans le répertoire **init.d**.

La présence ou l'absence d'un lien particulier détermine si le script est lancé ou non. De plus, la distinction entre les scripts d'entrée (*dont le lien commence par **'S'***) et les scripts de sortie (*dont le lien commence par **'K'***), associée à l'ordre induit par les deux chiffres, permet d'avoir un contrôle très précis sur la mise en place des services et leur arrêt.

Mais prenons un exemple. Dans le répertoire **rc5.d** se trouve un lien dont le nom est **S11httpd**. Le '**S**' nous indique qu'il s'agit d'un script qui sera lancé lors de l'entrée dans le niveau **5**, et sera (*probablement*) le onzième script à être exécuté. Le script sera **httpd** (*avec l'argument **'start'***) qui, en l'occurrence, met en place le serveur Web.

Ce lien possède un lien symétrique, **K03httpd**, qui sera exécuté lors de la sortie du niveau **5** en troisième position, le script étant toujours **httpd** mais, cette fois avec l'argument '**stop**'.

4. Contrôle et adaptation du démarrage

Il se peut que vous souhaitiez modifier les scripts lancés au démarrage ou plus généralement, créer une configuration radicalement différente de celle que vous utilisez quotidiennement (*par exemple, pour s'adapter à un environnement réseau*). Plusieurs méthodes sont à votre disposition pour ce faire.

La première, la plus mauvaise, consiste à modifier les scripts contenus dans le répertoire **/etc/rc.d/init.d**. Ceci est vivement déconseillé. Mieux vaut créer vos propres scripts pour vos besoins. Dans ce cas, placez-les dans le répertoire **/etc/rc.d/init.d**, afin de correspondre à la norme en vigueur.

Vous pouvez également manipuler les liens contenus dans les répertoires **rcn.d**, pour en rajouter ou en supprimer. Prenez simplement garde à l'ordre dans lequel ils doivent être lancés en fixant convenablement les deux chiffres contenus dans le nom du lien. De plus, pour tout script de démarrage ajouté (*commençant par **'S'***), il est recommandé de créer son alter ego pour l'arrêt (*commençant par **'K'***). Une bonne idée est de regarder de quelle façon sont faits les scripts déjà existants pour reprendre le même canevas. Notez qu'il existe un outil graphique pour ce genre d'opération, **tksysv** (*ou **ksysv**, si vous utilisez l'interface graphique **KDE***); il vous permet d'enlever ou d'ajouter des scripts dans les niveaux d'exécution et d'en préciser l'ordre d'exécution.

20 **S**: pour start ou lancer/démarrer.
21 **K**: pour kill ou arrêter.

Une autre possibilité est d'utiliser un niveau inutilisé normalement par votre distribution (*dans notre exemple, le niveau **4***) pour en faire ce que vous voulez. Si, de plus, vous placez un dièse ('**#**') au début de la ligne contenant le '**initdefault**' du fichier **/etc/inittab**, à chaque démarrage le système vous demandera quel niveau vous voulez; une bonne manière d'avoir plusieurs configurations selon les situations. En général, un niveau ainsi personnalisé est plus ou moins fortement inspiré d'un niveau existant. Il peut donc être intéressant de dupliquer un niveau de référence comme base de départ.

Pour ce faire, placez-vous dans le répertoire **/etc/rc.d** et exécutez:

```
rm rc4.d/*
```

pour "faire le ménage", c'est-à-dire "vider" le niveau **4**, et ainsi éviter l'influence de configurations précédentes. Puis, pour dupliquer par exemple le niveau **5**:

```
cp rc5.d/* rc4.d
```

Le niveau **4** est alors absolument identique au niveau **5** et c'est à vous de jouer pour modifier ce qui doit l'être...

Naturellement, les numéros indiqués ici ne sont que des exemples; adaptez-les pour correspondre à votre propre configuration.

5. Montage automatique des partitions: /etc/fstab

Le fichier **fstab** contenu dans le répertoire **/etc** est utilisé lors du démarrage du système pour déterminer quelles sont les partitions qui doivent être montées lors du démarrage et, plus généralement, donne des informations sur les différents systèmes de fichiers auxquels le système peut accéder. Considérons l'exemple suivant:

```
...
/dev/hdb5       /               ext2    defaults          1 1
/dev/hdb9       /home           ext2    defaults          1 2
/dev/hdb11      /home/ftp       ext2    defaults          1 2
/dev/hdb10      /home/httpd     ext2    defaults          1 2
/dev/hdb8       /root           ext2    defaults          1 2
/dev/hda7       /usr            ext2    defaults          1 2
/dev/hdb12      /usr/src        ext2    defaults          1 2
/dev/hda6       swap            swap    defaults          0 0
/dev/hda1       /mnt/dos1       vfat    defaults          1 2
/dev/fd0        /mnt/floppy     auto    user,noauto       0 0
/dev/cdrom      /mnt/cdrom      iso9660 user,noauto,ro    0 0
none            /proc           proc    defaults          0 0
...
```

Décrivons les différentes colonnes qui apparaissent.

La première colonne est manifestement le fichier spécial correspondant à la partition ou au périphérique concerné. Le "**none**" de la dernière ligne ne doit pas vous étonner. En fait, le répertoire **/proc** est utilisé de façon interne par le noyau (***kernel***). Il ne contient par réellement des fichiers. Ceux qui peuvent y apparaître ne sont que des images d'informations contenues dans le noyau.

La deuxième colonne est évidemment le point de montage (*le répertoire sur lequel on va accrocher la partition*). Remarquez la ligne contenant le mot "**swap**" dans la deuxième colonne. Cette ligne désigne en fait la partition de mémoire virtuelle qui ne se monte pas (*voir le paragraphe* ***La mémoire virtuelle*** *à la page* ***36*** *pour plus de détail concernant la mémoire virtuelle*).

La troisième colonne est le type de système de fichiers contenu sur la partition ou le périphérique concerné. "**ext2**" est le système **Ext2Fs**, utilisé normalement par Linux. "**vfat**" est le système de fichiers de Windows 95/98; "**iso9660**" est utilisé par les **CD-ROM**s. Notez "**auto**", dans la ligne commençant par **/dev/fd0**, qui correspond au lecteur de disquette. En effet, une disquette peut être formatée selon différents systèmes de fichiers. "**auto**" demande au système de deviner quel est le type présent sur une disquette que l'on voudrait monter.

Suivent quelques options pour préciser de quelle manière la partition doit être traitée. Le mot "**defaults**" indique qu'il faut appliquer les options par défaut; ce qui inclus entre autre le montage automatique. Toutes les partitions ainsi marquées seront montées avec la simple commande:

```
mount -a
```

Cette commande est exécutée durant le démarrage du système.

Par contre, les lignes commençant par **/dev/fd0** et **/dev/cdrom** comportent un "**noauto**". Ces périphériques ne seront pas montés automatiquement. Ce qui est, après tout, normal pour le lecteur de disquettes et le lecteur de **CD-ROM**s... Dans ces lignes apparaît également le mot "**user**" qui indique qu'un utilisateur quelconque est autorisé à monter le périphérique en question. Normalement, seul **root** est autorisé à monter les systèmes de fichiers. Consultez la page **man** de **mount** pour plus de détails concernant ces options.

L'avant-dernière colonne comporte un chiffre, **0** (*zéro*) ou **1**. La fonction exacte de ce chiffre est un peu obscure. Remarquez toutefois qu'il n'est à **1** que pour les systèmes de fichiers effectivement montés au démarrage.

Enfin, la dernière colonne indique dans quel ordre les systèmes de fichiers doivent être vérifiés (*c'est-à-dire contrôler qu'ils sont intègres et en bon état*) lors du redémarrage. Normalement, le système de fichier abritant le répertoire racine doit toujours être vérifié en premier. D'où le **1** dans la première ligne. Les autres peuvent être vérifiés dans n'importe quel ordre, mais après la racine, ils sont tous à **2**. Un **0** (*zéro*) indique que le système de fichiers n'a pas besoin d'être vérifié. C'est le cas naturellement pour la disquette et le **CD-ROM** mais, aussi pour la mémoire virtuelle et le "**pseudo-système de fichiers**" contenu dans **/proc**.

VII- Commandes "avancées"

1. Description générique de l'utilisation d'une commande

Lorsque l'on souhaite décrire de manière concise l'utilisation d'une commande, on utilise un certain formalisme: on parle alors du format de la commande. Par exemple, pour la commande **ln**:

```
ln [-bdfinsvF] [-S backup-suffix] [-V {numbered,existing,simple}] [-version-
control={numbered,existing,simple}] [-backup] [-directory] [-force] [-interactive] [-no-
dereference] [-symbolic] [-verbose] [-suffix=backup-suffix] [-help] [-version] source [dest]
```

Une autre forme parfaitement équivalente est:

```
ln [options] source [dest]

Options: [-bdfinsvF] [-S backup-suffix] [-V {numbered,existing,simple}] [-version-
control={numbered,existing,simple}] [-backup] [-directory] [-force] [-interactive] [-no-
dereference] [-symbolic] [-verbose] [-suffix=backup-suffix] [-help] [-version]
```

L'exemple choisi est particulièrement complexe, mais il offre tous les symboles que vous pouvez rencontrer.

Premier principe: tout ce qui est entre crochets (*'[' et ']'*) est facultatif. C'est-à-dire qu'il n'est pas obligatoire de donner l'option en question à la commande. Par exemple, dans ce qui précède, **-backup** est facultatif. Son absence n'empêchera pas la commande de s'exécuter, mais sa présence modifiera son fonctionnement. A l'inverse, ce qui n'est pas entre crochets est obligatoire, c'est le cas de **source**.

Deuxième principe; ce qui est entre accolades (*'{' et '}'*) présente un choix, en général exclusif. Par exemple, pour l'option **-version-control=** doit être suivie de l'un des mots **numbered**, **existing** ou **simple** (*ainsi que l'option **-V**, d'ailleurs*).

Ensuite, prenons le cas du premier crochet, **-bdfinsvF**. Ceci n'est manifestement pas un mot. Simplement, il est possible de donner aucune ou plusieurs options à la commande parmi '**-b**', '**-d**', '**-f**', etc et même des choses du genre '**-dnsF**' qui sont un collage de quatre options "**mono-lettre**".

Enfin, pour les options '**-S**' ou '**-suffix=**', le mot **backup-suffix** doit être compris comme devant être remplacé par une chaîne de caractères de votre choix. Ceci vaut également pour les mots source et **dest** (*qui en l'occurrence désignent des noms de fichiers*). En général, la description textuelle de la commande permet de savoir (*plus ou moins...*) quoi mettre à la place de ces mots à remplacer.

Selon ce modèle, la commande suivante est valide.

```
ln un_fichier.html un_autre_fichier.html
```

Voyez comment l'élément **source**, obligatoire pour cette commande, apparaît sous la forme de noms de fichiers.

Dans ce qui suit, nous utiliserons ce modèle pour décrire les commandes.

2. Retour sur la commande ls

La commande **ls**, avec l'option **-l**, permet d'obtenir les droits des fichiers d'un répertoire ainsi que de nombreuses autres informations. Considérons l'exécution de la commande '**ls -l**' dans un répertoire maison:

```
[root@ns-9 rep-maison]# ls -l

total 40
drwxr-xr-x   2    yves  users   1024   Nov 22  23:08  Ada
drwxr-xr-x   2    yves  users   1024   Dec 13  19:48  Courrier
-rw-r-r---   1    yves  users  13312   Jan 24  02:32  Curr_Vit.sdw
drwx------   5    yves  users   1024   Nov 27  21:28  Desktop
drwx------   2    yves  users   1024   Dec  9  10:58  Docs
drwxr-xr-x   2    yves  users   1024   Feb  1  19:13  Downloads
drwx------   2    yves  users   1024   Jan  3  22:52  Mail
drwxr-xr-x   21   yves  users  1024   Nov 23  21:36  Office50
drwx------   2    yves  users   1024   Nov 22  23:09  Progs
-rw-r--r--   1    yves  users    362   Feb  7  18:08  Xrootenv.0
drwxr-xr-x   9    yves  users   1024   Dec 28  17:38  YvesWeb
-rw-r--r--   1    yves  users   9860   Feb  2  19:08  cartes_visite.fig
-rw-r--r--   1    yves  users   5561   Nov 22  23:15  config-kernel
-rw-r--r--   1    yves  users   2643   Jan 13  22:55  config_kernel.txt
lrwxrwxrwx   1    yves  users     13   Mar 29  20:28  Doc_Linux -> LUG/Linux_doc
[root@ns-9 rep-maison]#
```

Décrivons les informations affichées.

La première ligne (*celle commençant par **"total"***) totalise l'espace disque occupé par les fichiers listés, en kilo-octets.

Ensuite vient la liste des fichiers, un par ligne. Chaque ligne se décompose en plusieurs colonnes:

- d'abord le type de fichier. La lettre '**d**' indique qu'il s'agit d'un répertoire, la lettre '**l**' signale un lien symbolique, un tiret '**-**' signale un fichier normal.
- ensuite les droits du fichier: de gauche à droite, on trouve les droits pour le **propriétaire**, puis ceux du **groupe**, puis ceux accordés au **reste du monde**. Ainsi, le fichier **config_kernel** est lisible par tout le monde, mais modifiable seulement par son propriétaire. Le répertoire **Docs** n'est modifiable, listable et "traversable" que par son propriétaire.
- puis le nombre de liens physiques sur le fichier (*voir le paragraphe **Mécanisme des liens** à la page **42** sur les liens pour plus de détails*). Pour un répertoire, cela équivaut grosso modo au nombre de fichiers qu'il contient (*rappelons que les répertoires sont considérés comme des fichiers*).
- vient alors le nom du **propriétaire** du fichier suivi du **groupe** auquel il appartient.
- puis l'**espace disque** occupé par le fichier, donné en octets.
- suivent la date et l'heure de la dernière modification du fichier (*attention! ce n'est pas forcément le dernier accès, si celui-ci était simplement une consultation*).
- et enfin le nom du fichier. Notez la dernière ligne, qui est un lien symbolique. Le nom de fichier est suivi des caractères '**->**' qui symbolisent une flèche. Puis le fichier auquel ce lien fait référence (*voir le paragraphe **Mécanisme des liens** à la page **42**, pour plus de détails*).

Les fichiers cachés (*commençant par un point '**.**'*), et notamment les répertoires spéciaux '**.**' et '**..**', n'apparaissent pas: il aurait fallu pour cela utiliser l'option '**-a**' de la commande **ls**.

3. Archivage de fichiers: tar et gzip

L'archivage est l'opération consistant à sauvegarder, d'une manière ou d'une autre, des fichiers importants. Soit que l'on veuille garder une trace d'un état ancien, soit que la perte de ces fichiers serait dramatique et donc on préfère en avoir une copie quelque part (*de préférence sur un support assez peu sensible à l'usure, comme un **CD** ou une bande*).

L'alter ego de l'archivage est la restauration qui est l'opération exactement inverse. À partir d'une sauvegarde, on récupère les fichiers qu'elle contient.

Il est rare de n'avoir qu'un seul fichier à sauvegarder. Plutôt quelques dizaines ou centaines et souvent l'espace occupé par ceux-ci est important. C'est pourquoi deux commandes furent créées pour faciliter cette opération: **tar** et **gzip**. D'autres commandes ou mécanismes plus sophistiqués existent, naturellement, mais ces deux commandes sont les plus répandues et toujours disponibles.

3.1. La commande tar

Elle permet, en gros, de "coller" plusieurs fichiers les uns au bout des autres pour n'obtenir qu'un seul fichier plus facilement manipulable. Toute sa puissance réside dans le fait qu'elle peut sauvegarder toute une arborescence. Si vos fichiers sont organisés en différents répertoires et sous-répertoires, la commande **tar** va "descendre" dans les répertoires qu'elle rencontre tout en conservant la structure qui sera remise en place lors d'une restauration éventuelle. Notez que **tar** était initialement destinée à être utilisées sur des bandes magnétiques (*des unités de stockages remarquables par leur capacité, leur durée dans le temps, et leur lenteur). Donc, si vous voulez créer des archives sous forme de fichiers sur votre disque dur, il faudra utiliser l'option **-f** (décrite plus bas*).

Les options de **tar** sont nombreuses. Le format de la commande est:

```
tar {c,d,r,t,u,x,A} [options] [fichier_tar] [fichiers...]
```

Ceci est évidemment un format abrégé. Normalement, **fichier_tar** est un fichier spécial (*du répertoire **/dev***) qui correspond à un lecteur de bandes, sauf à utiliser l'option **-f**.

La première lettre désigne l'action à effectuer:

c

Création d'une nouvelle archive.

d

Compare les fichiers contenus dans **fichier_tar** avec les fichiers indiqués dans **fichiers...**

r

Ajoute les fichiers **fichiers...** à la fin de l'archive.

t

Affiche les noms des fichiers **fichiers...** s'ils sont contenus dans l'archive. Si le paramètre **fichiers...** est absent, liste les fichiers contenus dans l'archive.

u

Mise à jour. Les fichiers **fichiers...** sont intégrés à l'archives s'ils ne s'y trouvent pas déjà ou bien s'ils ont été modifiés.

x

Restauration des fichiers **fichiers...** depuis l'archive **fichier_tar**. Si **fichiers...** est absent, restaure tous les fichiers.

A

Permet de coller deux fichiers archives.

3.1.1. Quelques options, parmi les plus utilisées

-f *fichier*

Sans doute l'option la plus utilisée. Permet d'utiliser un fichier sur disque, plutôt que des données sur bande, pour les opérations d'archivage (*sauvegarde ou restauration des fichiers*).

-h

Déréférence les liens symboliques. C'est-à-dire, ne stocke pas le lien en tant que lien, mais plutôt le fichier sur lequel il pointe.

-ignore-failed-read

Normalement, la commande **tar** s'arrête avec un message d'erreur si elle rencontre des fichiers qu'elle ne peut lire (*pour une raison ou pour une autre*). Ce paramètre permet de continuer l'exécution de **tar** dans ce cas.

-k

Lors de l'extraction, empêche d'écraser des fichiers existants par des fichiers contenus dans l'archive et ayant le même nom.

-v

Mode "verbeux". Affiche le nom des fichiers sauvegardés ou restaurés.

-w

Demande d'une confirmation avant d'effectuer une quelconque action.

-z

Compresse les fichiers avec **gzip** avant de les sauvegarder ou les décompresse lors de la restauration. Permet donc d'obtenir une archive de taille réduite. Toutefois, l'algorithme de **gzip** est tel qu'il est plus efficace de compresser le fichier archive en un bloc plutôt que chacun de ses fichiers.

-directory=rep

Demande à **tar** de se placer dans le répertoire **rep** avant de faire quoi que ce soit.

-M

Pour créer des archives en plusieurs morceaux; utile pour un stockage sur disquettes. À utiliser avec l'option **-L**.

-P

Normalement, le slash initial ('/') des noms de fichiers est supprimé par **tar**. Ce paramètre permet de stocker l'intégralité du chemin absolu de chaque fichier. Attention, ceci peut être dangereux lors de la restauration.

-L=taille

Défini la taille en kilo-octets de chaque morceau de l'archive. À utiliser avec **-M**.

La complexité de la commande justifie de donner ici quelques exemples. Remarquez que nous utilisons systématiquement l'option **-v** pour avoir une liste de fichiers qui s'affiche à l'écran. Ceci n'est toutefois pas une obligation.

- Utilisation "normale". Sauvegarde des répertoires **/home**, **/root** et de tous leurs sous-répertoires sur la bande qui se trouve dans le lecteur référencé par **/dev/rmt0**. Affiche le nom (*absolu*) de chaque fichier ainsi archivé. Notez la manière donc les commandes et les options sont accolées les unes aux autres. Gardez à l'esprit que le **slash** initial est supprimé.

```
tar cvf /dev/rmt0 /home /root
```


- Obtenir la liste des fichiers contenus dans l'archive qui se trouve sur la bande dans le lecteur **/dev/rmt0**.

```
tar tvf /dev/rmt0
```

- Sauvegarder le répertoire courant, en compressant les fichiers, et placer l'archive dans le fichier **../backup.tgz** (*l'extension **.tgz** est l'extension usuelle des archives dont les fichiers ont été compressés*) situé dans le répertoire parent.

```
tar zcvf ../backup.tgz .
```

- Restaurer les fichiers (*toujours dans le répertoire courant*) contenus dans l'archive **backup.tgz**.

```
tar zxvf backup.tgz
```

4. La commande gzip

La commande **gzip**[22], quant à elle, permet de compresser un ou plusieurs fichiers. Cela consiste à appliquer un certain algorithme sur le contenu du fichier pour obtenir à la fin, un fichier de taille inférieure. Ceci permet d'éviter trop de perte d'espace disque. Dans le cas de fichiers textes (*pages **HTML**, documents de traitement de textes...*), la compression est souvent très forte. Il est courant de voir la taille du fichier ramenée à un quart de la taille initiale. Par contre, un algorithme de compression appliqué à des fichiers déjà compressés ne permet pas de réduire leur taille, voire l'augmente légèrement... C'est notamment le cas des fichiers images dans certains formats, par exemple **GIF** ou **JPEG**. Notez que le résultat d'une compression appliquée à un fichier a pour résultat un nouveau fichier de taille certes inférieure, mais parfaitement illisible pour un œil humain. Et d'une manière générale, plus le fichier à compresser est volumineux, plus le taux de compression (*le rapport entre la taille initiale et la taille après compression*) est élevé.

L'opération inverse est évidemment possible. À partir d'un fichier compressé, on récupère alors le fichier dans sa forme initiale sans aucune perte d'information (*encore que ceci dépende des algorithmes; celui utilisé par gzip ne provoque pas de perte d'information*).

L'utilisation générale de **gzip** est de la forme:

```
gzip [options] [fichiers]
```

4.1.1. Quelques options

-n

Où **n** est un nombre de **1** à **9**. Permet de spécifier, en quelque sorte, la puissance de la compression. **1** est une compression minimum et rapide, **9** tente de compresser au mieux et donc plus lentement.

-d

Pour décompresser un fichier compressé.

-r

Lorsque que **fichiers** correspond à un répertoire, cette option permet de compresser les fichiers et les sous-répertoires qu'il contient.

-S *suffix*

Précise quelle extension doit être utilisée (*par défaut, c'est **.gz***). Utile lorsque l'on tente de décompresser des fichiers compressés avec d'autres utilitaires compatibles tels que **pkunzip** sous DOS ou **WinZip** sous Windows (*qui créent une extension **.zip***).

22 **gzip**: pour "**G**NU **Zip**", la version **GNU** de l'algorithme de compression de Lempel-Ziv.

4.1.2. Quelques exemples

- Compresser au mieux tous les fichiers du répertoire courant. Ces fichiers disparaîtront et seront remplacés par des fichiers portant le même nom, mais avec l'ajout de l'extension **.gz**.

```
gzip -9 *
```

- Compresser au mieux le fichier **backup.tar** qui deviendra alors **backup.tar.gz** et affiche le taux de compression.

```
gzip -9 -v backup.tar
```

- Décompresser le fichier **backup.tar.gz**.

```
gzip -d backup.tar.gz
```

- Décompresser le fichier **images.zip**.

```
gzip -d -S zip images.zip
```

Il semble évident que, dans l'optique de sauvegarder des fichiers, l'utilisation conjointe de **tar** et de **gzip** offre des possibilités intéressantes. Typiquement, une utilisation conjointe avec transfert du fichier archive sur une autre partition ou sur un réseau serait quelque chose du genre (*pour sauvegarder le répertoire **/home***):

```
tar cvf backup.tar /home
```

```
gzip -9 backup.tar
```

(O*n utilise **cp** plutôt que **mv**, dans le cas ou la copie se passe mal*).

```
cp backup.tar.gz /mnt/nfs
```

(*Inutile de garder l'archive à son point de création puisqu'elle a été copiée*).

```
rm backup.tar.gz
```

Par la suite, les fichiers ainsi sauvegardés pourront être restaurés par les opérations:

```
cp /mnt/nfs/backup.tar.gz .
```

```
gzip -d backup.tar.gz
```

```
tar xvf backup.tar
```

5. Ré-attribution d'un fichier: chown, chgrp

5.1. Modifier les droits sur un fichier ou un répertoire: chmod

Cette commande permet de modifier les droits (*les attributs*) d'un fichier. Si vous ne connaissez pas la notion de droits sur un fichier, consultez le paragraphe ***Droits des fichiers et des répertoires*** à la page ***43***. La première chose à bien comprendre est que vous ne pouvez naturellement modifier les droits que d'un fichier dont vous êtes le propriétaire sinon, Linux ne serait pas un système très sécurisé...

chmod reconnaît les abréviations **r**, **w** et **x** pour les droits (*respectivement, lecture, écriture et exécution*) et les abréviations **u**, **g** et **o** pour les groupes de droits (*respectivement, groupe relatif à l'utilisateur propriétaire, au groupe propriétaire, ou "aux autres*[23]*"*).

23 **r** pour Read, lecture; **w** pour Write, écriture; **x** pour eXecute, exécution.

Sous réserve que vous en ayez le droit, vous pouvez ajouter ou retirer un ou plusieurs droits d'un ou plusieurs groupes de droits, en indiquant les abréviations des groupes concernés puis, un signe plus ('+') pour ajouter ou un signe moins ('-') pour retirer puis, les abréviations des droits que vous voulez modifier.

Par exemple, pour accorder les droits en lecture et écriture pour le propriétaire et le groupe propriétaire sur le fichier **Rapport.lyx**, exécutez la commande:

```
chmod ug+rw Rapport.lyx
```

Notez que vous pouvez également appliquer ceci sur des répertoires.

Il est également possible d'utiliser les caractères **jokers**. Ainsi, pour accorder à vous seul le droit d'exécution sur tous les fichiers d'extension '**.x**' du répertoire courant, exécutez:

```
chmod u+x *.x
```

Enfin, l'option **-R** permet d'appliquer **chmod** récursivement sur l'arborescence contenue dans le répertoire.

```
chmod -R u+x programmes/*.exe
```

Ceci va fixer l'attribut d'exécution sur tous les fichiers d'extension '**.exe**' dans le répertoire programmes et dans tous les sous-répertoires qu'il peut contenir.

6. Manipulation des liens: ln

Si vous ne l'avez déjà fait, consultez le paragraphe ***Mécanisme des liens*** à la page ***42*** pour bien comprendre la notion de lien sous Linux. La commande présentée ici permet de créer des liens, symboliques ou non. Sa forme générale est:

```
ln [options]  fichier_à_lier  fichier_lien
```

L'option la plus utile est l'option **-s**, qui permet de créer des liens symboliques plutôt que des liens physiques. Voici un exemple typique d'utilisation de **ln**.

Créer le fichier **/cdrom**, qui sera en fait un lien symbolique vers le répertoire **/mnt/cdrom**.

```
ln -s /mnt/cdrom /cdrom
```

Par la suite, vous pourrez indifféremment utiliser l'un ou l'autre dans vos commandes, par exemple:

```
mount /dev/hdd /cdrom
```

```
mount /dev/hdd /mnt/cdrom
```

Ces commandes sont parfaitement équivalentes.

7. Montage et démontage de systèmes de fichiers: mount et umount

Ces deux commandes sont spécialement destinées au montage et au démontage des systèmes de fichiers. Deux opérations essentielles décrites au paragraphe ***notion de montage*** à la page ***40***.

7.1. mount

La commande **mount** permet de monter un système de fichier en un point de l'arborescence. Sa forme est:

```
mount [options] [fichier spécial] [point de montage]
```

Si aucun paramètre n'est donné à la commande, on obtient simplement la liste des systèmes de fichiers actuellement montés. En principe, dès qu'un paramètre est donné, seul **root** peut utiliser **mount**.

L'utilisation la plus commune de **mount** est:

```
mount /dev/hdd /mnt/cdrom
```

Ceci monte simplement le système de fichiers présent sur le **CD-ROM** dans le lecteur, en l'accrochant au répertoire **/mnt/cdrom**. Notez qu'il est tout à fait possible, sur votre système, que le fichier spécial impliqué ne soit pas **/dev/hdd**. En principe, sous cette forme, vous verrez un message vous avertissant que le système de fichiers est monté en "lecture seule". Cette petite complainte est normale, on ne peut écrire sur un **CD-ROM**.

Par ailleurs, la commande précédente peut échouer et renvoyer une erreur dans l'un des cas suivants:

- il n'y a pas de **CD-ROM** dans le lecteur;
- le répertoire courant d'au moins un utilisateur connecté au système est justement **/mnt/cdrom** ou l'un de ses sous-répertoires. Dans ce cas, **mount** refuse d'effectuer le montage;
- le système de fichier qui se cache derrière **/dev/hdd** est déjà monté quelque part. En effet, on ne peut monter plusieurs fois un même système de fichiers.

Si la commande échoue, c'est le plus souvent pour l'une des raisons précédentes.

La commande **mount** ne possède que peu de paramètres.

-a

Tente de monter tous les systèmes de fichiers indiqués dans **/etc/fstab** qui ne présentent pas l'option "**noauto**". Dans ce cas, il n'est pas nécessaire de préciser un fichier spécial ni un point de montage (*puisqu'ils sont déjà indiqués dans le fichier **/etc/fstab***).

-n

Normalement, lorsque **mount** monte un système de fichier, celui-ci est noté dans le fichier **/etc/mtab**. Ce paramètre inhibe cette action.

-t *<type>*

Indique à **mount**, le type de système de fichiers que l'on tente de monter. Ceci ne devrait pas être nécessaire car, en général, **mount** devine assez bien à quoi il a affaire. Les types les plus courants sont:

msdos

Pour les anciens systèmes MS-DOS.

vfat

Pour les systèmes Windows 95 et Windows 98.

ntfs

Pour le système de Windows NT/Win-7/8.x/10.

iso9660

Pour les **CD-ROM**s, du nom de la norme **ISO** qui leur est réservée.

hpfs

Pour le système OS/2.

[ext2 | ext3 | ext4 | reiserfs]

Pour le système natif de Linux.

nfs

Pour les systèmes de fichiers exportés sur le réseau, selon la norme **NFS**.

-o ***<options>***

Options complémentaires pour affiner le montage. Nombre d'entre elles sont spécifiques d'un certain type de système de fichiers mais, certaines sont communes à tous, telles que:

noexec

Pour interdire l'exécution de programmes présents sur ce système de fichiers.

ro

Protège le système de fichier contre l'écriture.

rw

Autorise lecture et écriture.

remount

Utilisée sur un système déjà monté, pour en modifier les options.

-v

Affiche plein de messages d'informations lors du montage.

7.2. umount

La commande **umount** effectue l'opération inverse, c'est-à-dire démonte le système de fichiers. En reprenant l'exemple précédent, son utilisation usuelle peut prendre l'une des deux formes suivantes, parfaitement équivalentes:

```
umount /dev/hdd
```

```
umount /mnt/cdrom
```

On peut donc spécifier indifféremment le fichier spécial ou le point de montage. La commande peut échouer dans l'un des cas suivants:

- un fichier ou un répertoire sous le point de montage est en cours d'utilisation,
- rien n'est monté au point spécifié,
- le fichier spécial indiqué n'a pas fait l'objet d'un montage.

umount ne reconnaît que les trois paramètres suivants, pratiquement jamais utilisés:

-a

tente de démonter tous les systèmes de fichiers notés dans **/etc/mtab**;

-n

à l'instar de **mount**, lorsque **umount** démonte un système de fichiers, celui-ci est retiré du fichier **/etc/mtab**; ce paramètre inhibe cela.

-t ***<type>***

ne démonte que les systèmes de fichiers du type indiqué.

8. Entrées/sorties directes de données: dd

Cette commande est l'une des plus dangereuses qui soit. Elle permet d'écrire d'un endroit à un autre une série d'octets, sans s'occuper d'une éventuelle structuration de là où elle écrit. Vous pouvez ainsi, en une seule ligne, faire totalement et irrémédiablement disparaître les partitions de votre disque dur et rendre celui-ci inutilisable...

Sa forme générale est la suivante:

```
dd [-help] [-version] [if=file] [of=file] [ibs=bytes] [obs=bytes] [bs=bytes] [cbs=bytes]
[skip=blocks] [seek=blocks] [count=blocks]
[conv={ascii,ebcdic,ibm,block,unblock,lcase,ucase,swab,noerror,notrunc, sync}]
```

L'utilisation la plus fréquente que vous aurez sans doute à en faire est de prendre l'image binaire d'une disquette. Par image binaire, on entend habituellement recopier dans un fichier le contenu exact de la disquette, sans tenir compte de la manière dont elle est structurée ou formatée, sans savoir si elle contient ou non des fichiers. La commande est généralement:

```
dd if=/dev/fd0 of=image.bin
```

Ceci provoque la lecture par **dd** de tout ce qui se trouve derrière **/dev/fd0** (*qui correspond au premier lecteur de disquette*), sans opérer un quelconque montage de système de fichiers. L'écriture se fait dans le fichier **image.bin** dans le répertoire courant. Le fichier obtenu sera de la taille de la disquette en tant que support "brut", c'est-à-dire, légèrement supérieure à l'espace normalement disponible sur une disquette.

Particulièrement utile lorsque l'on souhaite copier une disquette sans s'occuper du type des fichiers qui y sont inclus ou bien lorsque le système de fichiers qu'elle contient ne peut être reconnu.

Les paramètres de **dd** les plus utilisés sont:

--help

Donne un cours message d'aide;

--version

Donne la version de **dd** installée;

if=file

Pour "input file" (*fichier d'entrée*). Indique la source que **dd** doit utiliser. Par défaut, c'est l'entrée standard, i.e le clavier.

of=file

Pour "output file" (*fichier de sortie*). Indique vers quoi **dd** doit inscrire les octets qu'elle a lue. Par défaut, c'est la sortie standard, i.e. l'écran.

ibs=bytes

Pour utiliser une lecture par blocs d'octets. Indiquer la taille des blocs (*en nombre d'octets à lire en une seule fois*).

obs=bytes

Pour utiliser une écriture par blocs d'octets. Indiquer la taille des blocs (*en nombre d'octets à écrire en une seule fois*).

bs=bytes

Combine les deux paramètres précédents, si l'on veut des blocs de même taille en lecture et en écriture.

skip=blocks

Indique combien de blocs il faut "sauter" avant de commencer à lire. La taille en octets de ces blocs est définie par **ibs** ou **bs**.

seek=blocks

Indique combien de blocs il faut "sauter" avant de commencer à écrire. La taille en octets de ces blocs est définie par **obs** ou **bs**.

count=blocks

Indique combien de blocs il faut lire. Par défaut, **dd** lit tous les blocs qu'elle trouve.

Donnons quelques exemples, pour éclairer un peu tout cela.

La commande suivante sauvegarde la **Table des Partitions** du premier disque dur **IDE** (*cette table se trouvant dans le premier secteur physique du disque*). La sauvegarde se fait dans le fichier **partitions.img**:

```
dd if=/dev/hda of=partitions.img bs=512 count=1
```

Le fichier résultant a une taille d'exactement 512 octets qui est la taille normale d'un secteur de disque. La **Table des Partitions** pourra par la suite être restaurée avec:

```
dd if=partitions.img of=/dev/hda
```

Les spécifications de taille sont superflues dans cette dernière commande.

9. Système de fichiers de Linux: mke2fs, ext2fsck

Il en est de Linux comme des autres systèmes. Pour qu'une partition puisse être utilisée avec le système de fichiers propre à Linux, il est nécessaire de la préparer. De plus, avec le temps, de petites erreurs ou incohérences peuvent se glisser dans la structure du système de fichier. Il est donc nécessaire de le vérifier régulièrement et éventuellement de corriger.

9.1. La commande mke2fs

Cette commande est utilisée pour créer un système de fichiers (*vide, sans aucune information à l'intérieur*) sur un disque ou une partition. Elle possède de nombreuses options, nous n'évoquerons ici que les plus fréquemment utilisées.

Son format complet est:

```
mke2fs [ -c | -l filename ] [ -b block-size ] [ -f fragment-size ]
[ -i bytes-per-inode ] [ -N number-of-inodes ]
[ -m reserved-blocks-percentage ] [ -o creator-os ] [ -q ]
[ -r fs-revision-level ] [ -R raid_options ] [ -s sparse-super-flag ]
[ -v ] [ -F ] [ -L volume-label ] [ -M last- mounted-directory ]
[ -S ] [ -V ] device [ blocks-count ]
```

9.1.1. Quelques options

-c

Recherche les blocs physiquement défectueux lors du formatage.

-i bytes-per-inode

Nombre d'octets dans chaque **i-node**. Doit être un multiple de 1024 (*1 kilo-octet*) et par défaut vaut 4096. Si vous estimez que le disque va contenir beaucoup de petits fichiers, une valeur de 2048 est un bon compromis. Par contre, si vous pensez stocker principalement de gros fichiers, préférez des valeurs de 8192, voire davantage.

-m reserved...

Pourcentage de blocs réservés pour le **super-utilisateur** (***root***), par défaut 5%. Pour une partition destinée uniquement au stockage de données, vous pouvez réduire ce nombre pour gagner un peu d'espace (*mais ne le fixer jamais à zéro*).

-v

Mode verbeux. Affiche des informations lors de l'exécution.

-L volume-label

Permet d'affecter un nom au disque ou à la partition (*ce qui est rarement utile...*).

Par exemple, vous voulez préparer une disquette pour être utilisée par Linux (*attention, elle ne sera plus lisible par d'autres systèmes!*), exécutez alors simplement:

```
mke2fs /dev/fd0
```

Pour formater une partition **/dev/sdb7** destinée à recevoir des fichiers de grandes tailles et nommée **BigDatas** (*avec détails des opérations*):

```
mke2fs -i 8192 -m 1 -v -L BigDatas /dev/sdb7
```

9.2. La commande e2fsck

Il est nécessaire de vérifier de temps à autre la cohérence d'un système de fichiers (*en général, se fait automatiquement à intervalles réguliers lors du démarrage du système*). Cette commande permet également de réparer (*dans une certaine mesure*) un système de fichiers endommagé. Ce qui peut survenir lors d'un plantage du système ou de son arrêt intempestif.

Son format complet est:

```
e2fsck [ -pacnyrdfvstFSV ] [ -b superblock ]
[ -B block- size ] [ -l|-L bad_blocks_file ] [ -C fd ] device
```

9.2.1. Quelques options

-b superblock

Permet d'utiliser un autre **superblock** que la normale. Peut aider dans certains cas. En général, des copies du **superblock** se trouve dans les blocs 8193, 16385, et ainsi de suite.

-c

Recherche des secteurs physiquement défectueux sur le disque.

-f

Force l'exécution, même si le système de fichiers semble correct.

-p

Réparation automatique.

-v

Mode verbeux. Affichage d'informations lors de l'exécution.

Il est fréquent, à la suite d'un plantage de la machine, que lors du démarrage, le processus d'initialisation s'interrompe pour vous signaler qu'un système de fichiers devant être monté (*par exemple, **/dev/hda5***) est probablement défectueux. Il vous est alors demandé d'entrer le mot de passe **root** puis, d'exécuter **e2fsk** manuellement (*attention, il est possible que votre clavier soit configuré en clavier américain*). Si l'arrêt intempestif de la machine est survenu lors d'une très faible activité du disque, la première commande à utiliser est alors:

```
e2fsck -f /dev/hda5
```

Si par contre, le disque connaissait une activité intense, il est probable que le **superblock** principal soit défectueux. Utilisez alors plutôt:

```
e2fsck -f -b 8193 /dev/hda5
```


Dans ce cas, vous pouvez également utiliser l'option **-c** si vous craignez pour l'intégrité physique du disque. L'exécution prend beaucoup plus de temps...

Il peut arriver, dans certains cas, que **e2fsck** ne parvienne pas à corriger certaines erreurs parce que trop complexes ou trop nombreuses. Utilisez alors la combinaison de la dernière chance:

```
mke2fs -S /dev/hda5
```

```
e2fsck -f /dev/hda5
```

Mais inutile de vous dire que dans ce cas, il n'est pas garanti que vous puissiez récupérer toutes vos données... Il est toutefois très rare d'en arriver à cette extrémité.

10. Chercher un fichier dans l'arborescence: find

La commande **find** est utilisée pour rechercher des fichiers dans une arborescence selon un ou plusieurs critères et éventuellement d'effectuer une action sur les fichiers qui correspondent aux critères donnés. Cette commande est particulièrement riche et complexe à utiliser aussi, n'en donnerons-nous ici qu'un aperçu général. Pour plus de détails, consultez la page **man** associée.

La forme générale est:

```
find [chemins] [critères] [actions]
```

chemins désigne une liste de répertoires à utiliser comme points de départ de la recherche. Si aucun n'est indiqué, le répertoire courant est considéré comme point de départ. Dans tous les cas, les éventuels sous-répertoires contenus dans les (*ou le*) point de départ sont également parcourus pour la recherche.

critères est une succession de conditions que les fichiers doivent satisfaire pour être retenus. Citons les plus utilisés:

-name "*modèle*"

donne un modèle pour le nom des fichiers. Il est possible d'utiliser les caractères **jokers * et ?** dans le modèle mais, dans ce cas, n'oubliez pas les guillemets;

-iname "*modèle*"

identique à **-name** mais, sans tenir compte de la casse des lettres (*c'est-à-dire ne différencie pas les majuscules des minuscules*);

-atime +n|-n|n

recherche les fichiers dont le dernier accès remonte à plus de, moins de, ou exactement **n** jours (*n'utilisez que l'une des trois formes*);

-mtime +n|-n|n

même chose mais, concernant la dernière modification plutôt qu'un simple accès au fichier.

-mount

restreint la recherche au même système de fichier que le point de départ;

-user *nom*

recherche les fichiers appartenant à l'utilisateur dont on donne le ***nom*** utilisé pour la connexion;

Enfin, **actions** est une série d'actions à effectuer sur les fichiers qui répondent aux critères donnés. Si aucune action n'est précisée, **find** affiche simplement le nom du fichier assorti de son chemin absolu complet. Citons:

-print

c'est l'action par défaut décrite plus haut;

-exec <command> {} \;

exécute la commande **<commande>** chaque fois qu'un fichier répondant aux critères est trouvé. Ce peut être **gzip**, **rm**, **more**, etc. La paire d'accolades (*"{ }"*) symbolise le nom du fichier. N'oubliez pas le "**\;**" à la fin;

-ok <command> {} \;

même chose que **-exec** mais, avec une demande de confirmation avant d'exécuter la commande.

Donnons quelques exemples:

• Obtenir la liste des fichiers ayant l'extension **.lyx** dans les répertoires **Travail** et **Ecriture**:

```
find Travail Ecriture -name ''*.lyx''
```

• Trouver tous les fichiers **core** à partir du répertoire courant et les effacer sans confirmation:

```
find -name core -exec rm {} \;
```

• Trouver tous les fichiers modifiés depuis les deux derniers jours:

```
find / -mtime -2
```

11. Manipulations sur les processus: ps, kill

Nous avons vu en début d'ouvrage, la notion de processus (*voir le paragraphe* ***Programme, processus, logiciel & Co*** *à la page* ***27***). Les deux commandes présentées ici permettent de les manipuler.

11.1. La commande ps

Elle permet d'obtenir des informations concernant les processus en cours d'exécution. Son format est:

```
ps [options]
```

Sans options, **ps** liste simplement les processus lancés par vous depuis votre connexion au système.

Les options reconnues par **ps** sont nombreuses, nous n'évoquerons ici que les plus courantes.

a

Liste tous les processus.

e

Inclue les variables d'environnement dans la description d'un processus.

f

Fait apparaître l'arborescence des processus.

l

Affichage long.

x

Affiche également les processus qui ne sont pas associés à une console particulière (*c'est généralement le cas pour les* ***daemons*** *ou les programmes fonctionnant sous interface graphique*).

Prenons un exemple:

```
[root@tchana ~]# ps xf

 PID  TTY   STAT  TIME  COMMAND
 708   ?     S    0:12  /usr/X11R6/bin/wmaker
 720   ?     SN   0:00  perl ./build_signature
 741   ?     S    0:00  wmclock -shape
 742   ?     S    0:02  kvt -T K Virtual Terminal
 745 ttyp1   S    0:00  \_ bash
1003 ttyp1   S    0:01     \_ kscd
6533 ttyp1   R    0:00     \_ ps xf
 953   ?     S    0:19  lyx
[root@tchana ~]#
```

On voit ici plusieurs processus n'ayant pas de console associée (*colonne **TTY***) et l'arborescence nous montre, par exemple, que le processus **kvt** (*de **PID 742***) a donné naissance au processus **bash** qui, lui-même, a créé les processus **kscd** et **ps** (*il est normal que **ps** apparaisse dans la liste qu'il affiche. Après tout, lors de son exécution, c'est également un processus!*).

Naturellement, tous les processus du système n'apparaissent pas ici. Le sens des colonnes est le suivant:

PRI

Priorité du processus. Un nombre élevé indique une priorité faible (*n'apparaît pas dans l'encadré ci-dessus*).

PID

Numéro unique affecté au processus par le système.

STAT

État du processus:

R

en cours d'exécution;

T

arrêté;

D

inactif et ne pouvant être interrompu;

S

inactif (endormi);

Z

zombie;

N

Valeur "**nice**" positive, c'est-à-dire processus a très faible priorité. (*n'apparaît pas dans l'encadré ci-dessus*).

TTY

Console associée au processus.

TIME

Temps accordé au processus sur le système, en heures et minutes.

Il existe d'autres colonnes, qui ne sont pas affichées ici. Consultez la (*longue*) page **man** de **ps** pour plus de détails.

Une utilisation fréquente de **ps** est de filtrer sa sortie avec un **grep**. Par exemple, si nous voulons obtenir tous les processus résultant de l'exécution du programme **httpd**, nous pourrions utiliser:

```
ps ax | grep httpd
```

Résultat:

```
 617   ?    S  0:00 httpd
 620   ?    S  0:00 httpd
 621   ?    S  0:00 httpd
 622   ?    S  0:00 httpd
 623   ?    S  0:00 httpd
6557 ttyp1  S  0:00 grep httpd
```

Le programme **httpd** a donc été ici lancé cinq fois.

ps nous permet donc d'obtenir des informations sur les processus. Voyons maintenant comment agir sur eux.

11.2. La commande kill

Elle permet d'envoyer des signaux aux processus. Le mécanisme des signaux est un des aspects fondamentaux du fonctionnement d'un système de type Unix. Il s'agit simplement d'attirer l'attention d'un processus et de lui donner un nombre. Selon la valeur de ce nombre, le processus réagi en conséquence si une procédure prévue pour ce signal est prévue. Les processus peuvent en effet tout simplement ignorer les signaux qu'ils reçoivent, sauf quelques-un.

Ce mécanisme peut paraître extrêmement simpliste, mais il est très utilisé dans la communication entre processus. Ceux-ci ont rarement beaucoup de choses à se raconter et un simple nombre est souvent suffisant.

Le format de **kill** est:

```
kill [-l | -nombre] PID
```

L'option **-l** donne une liste de signaux prédéfinis. Cette liste donne le numéro du signal, ainsi que le mot équivalent pouvant être utilisé à la place du nombre.

Notez qu'il est toujours nécessaire de donner le numéro du processus (*obtenu avec **p**, colonne **PID***) auquel s'adresse le signal.

11.2.1. Quelques signaux très utilisés

-1 ou -SIGHUP

Demande au processus de se "ré-initialiser", c'est-à-dire, en quelque sorte, de reprendre son exécution depuis le début. Utilisé essentiellement sur les **daemons** pour leur faire prendre en compte des modifications dans certains fichiers de configuration.

-15 ou -SIGTERM

Demande au processus de terminer son exécution, en exécutant immédiatement le code prévu à cet effet. C'est une manière "douce" de terminer un processus qui semble ne plus répondre.

-9 ou -SIGKILL

Ce signal ne peut être ignoré, que le processus ait prévu ou non une réponse. L'éventuel code de terminaison est exécuté si possible, puis le processus est vidé de la mémoire. C'est la manière "forte" pour faire disparaître un processus particulièrement récalcitrant.

Voici un exemple amusant. Connectez-vous au système sur une console texte quelconque, puis exécutez **ps**:

```
...
  PID TTY          TIME    CMD
 6585 tty2     00:00:00  bash
 6605 tty2     00:00:00  ps
...
```

Le **shell** actif est le **shell bash**. Envoyons-lui un signal avec:

```
kill -15 6585
```

Rien ne se passe... parce que le **bash** en question est le **shell** principal de la connexion. Il ignore donc le signal **15 (*SIGTERM*)**. Soyons plus énergique:

```
kill -9 6585
```

Cette fois, vous vous retrouvez devant l'écran de connexion comme si vous vous étiez déconnecté! La commande précédente "tue" littéralement le processus **bash** et comme c'était le processus qui vous permettait d'utiliser le système, vous vous trouvez déconnecté. Les deux prochaines commandes sont parfaitement équivalentes à celles-ci:

```
kill -SIGTERM 6585
```

```
kill -SIGKILL 6585
```

Prenez garde en utilisant les signaux (*surtout le **SIGKILL***), vous pourriez être obligé de redémarrer votre machine...

Pour conclure, sachez également que, normalement, un utilisateur ne peut pas faire ainsi disparaître les processus d'autres utilisateurs (*sauf bien sur, l'utilisateur **root**, qui a tous les droits...*).

12. Partitionnement du disque: fdisk

Ce petit utilitaire vous permet de manipuler les partitions d'un disque dur. De nombreux autres utilitaires du même genre existent, souvent, plus agréables à utiliser. Mais, **fdisk** est le seul qui soit (*normalement*) toujours disponible et suffisamment peu exigeant pour fonctionner dans pratiquement n'importe quelle situation. Si vous n'êtes pas familier des notions relatives aux partitions des disques, consultez le paragraphe ***Partition des disques durs*** à la page ***32***.

Le format de la commande pour l'appeler est:

```
fdisk [-l | -s partition] [disque]
```


12.1. Les options

-l

donne une liste des partitions des différents disques, attachés au système, avec leurs caractéristiques (*cylindres de début et de fin, taille, type de système de fichiers...*), par exemple:

```
[root@tchana ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0004df76

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    41943039    19921920   8e  Linux LVM

Disk /dev/mapper/VolGroup-lv_root: 18.2 GB, 18249416704 bytes, 35643392 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VolGroup-lv_swap: 2147 MB, 2147483648 bytes, 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@tchana ~]#
```

Cet exemple montre la présence de deux disques durs sur le premier port **IDE**. La colonne "**Id**" est celle du code du type de système de fichiers présent sur la partition. La colonne "**Boot**" indique si la partition est celle utilisée lors du démarrage du système, ici **/dev/hda2**. Remarquez que les tailles sont données en kilo-octets.

-s ***partition***

```
[root@tchana ~]# fdisk -s /dev/sda1

1048576
[root@tchana ~]#
```

disque

indique le disque dont on veut manipuler les partitions dans le mode interactif.

Le mode d'utilisation courant de **fdisk** est le mode interactif, si ni "**-l**" ni "**-s**" n'est donné. C'est-à-dire que le programme se comporte un peu comme un **shell** (*en beaucoup plus simple*), attendant que vous entriez une commande au clavier suivie de [**Entrée**]. Ces commandes sont de deux catégories: celles un peu dangereuses et celles très dangereuses. Citons les plus utilisées (*mais il en existe de nombreuses autres*):

m

affiche une liste des commandes disponibles.

p

affiche la liste des partitions avec leurs caractéristiques. Utilisez fréquemment cette commande, les numéros de partition peuvent varier au cours des manipulations.

d

supprime une partition dont on donne le numéro. Vous ne pouvez normalement pas supprimer une partition étendue si celle-ci contient encore des partitions logiques.

n

créé une nouvelle partition. Le programme vous demande d'abord quel type de partition vous voulez: primaire ('**p**'), étendue ('**e**') ou logique ('**l**'). Vous devez créer une partition étendue avant de pouvoir créer des partitions logiques. Par ailleurs, notez bien que vous ne pouvez créer qu'une seule partition étendue (*mais qui peut contenir presque autant de partitions logiques que vous voulez*).

Ensuite, vous est demandé le numéro de la partition. Rappelez-vous que Linux numérote les partitions primaires de 1 à 4 et les partitions étendues toujours à partir de 5.

Il vous faut maintenant donner le premier cylindre de la partition dans l'intervalle qui vous est proposé. Entrez la plus petite valeur, bien souvent, elle convient parfaitement. Puis le dernier cylindre. Il est souvent plus commode de donner la taille désirée de la partition. Par exemple, pour créer une partition de 327 Mo, entrez "**+327M**". Vous remarquerez peut-être un léger décalage entre les tailles demandées et les tailles réelles. Ceci vient de l'obligation de borner les partitions selon les limites de cylindres et il est probable que la taille que vous avez demandée ne soit pas un multiple de la taille d'un cylindre.

t

Par défaut, toute partition créée (*sauf les partitions étendues, bien sûr*) est marquée comme contenant le système de fichiers **Ext2Fs**, le système de fichiers de Linux. Cette commande permet de modifier cela. Le programme demande d'abord le numéro de la partition, puis le code du système de fichier voulu. Entrez '**l**' pour obtenir une liste des codes connus.

w

inscrit les changements sur le disque et quitte le programme. Si vous manipulez des partitions de type MS-DOS (*FAT16 ou FAT32*), il est possible qu'un message d'erreur survienne lorsque le programme se termine. Il suffit normalement de redémarrer la machine pour que les changements soient correctement pris en compte.

q

quitte **fdisk** sans enregistrer les changements sur le disque.

Gardez toujours présent à l'esprit que les manipulations des tables de partitions sont des opérations particulièrement sensibles. Prenez donc toujours garde à ce que vous faites.

Par ailleurs, il est déconseillé (*bien que cela soit possible*) d'utiliser **fdisk** pour manipuler des partitions autres que celles devant êtres utilisées exclusivement par Linux (*ceci concerne notamment les partitions DOS/Windows*). Mieux vaut utiliser le programme de partitionnement spécifique de l'autre système que vous utilisez. Dans le même ordre d'idée, essayez de faire en sorte que les partitions soient numérotées "dans l'ordre". Par exemple, **fdisk** vous permet d'insérer (*s'il y a de la place*) la partition n° 7 entre les partitions 5 et 6, en jouant sur le numéro de premier cylindre. Il faut absolument l'éviter, cela ne pose pas de problème à Linux, mais c'est loin d'être le cas de tous les systèmes. Donc faites attention si vous utilisez un autre système d'exploitation que Linux sur votre machine.

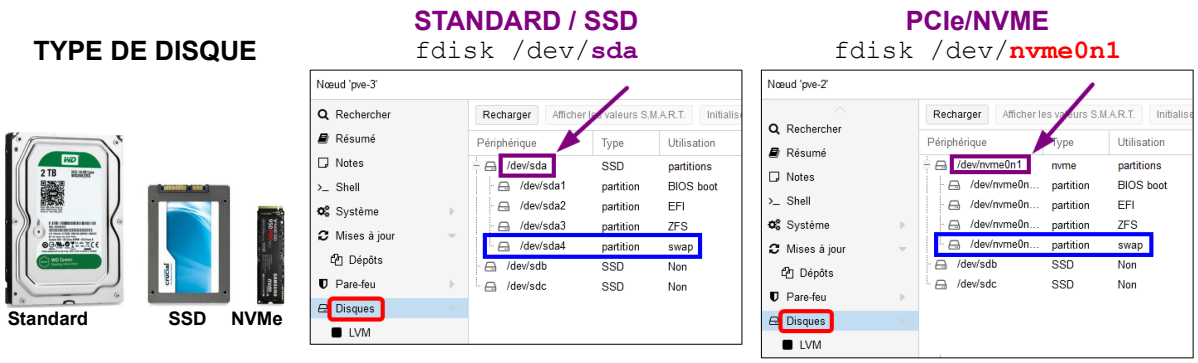
Enfin, vous aurez remarqué l'absence de commandes pour déplacer ou changer la taille d'une partition. **fdisk** ne permet pas ces opérations. Vous devez détruire une partition pour, par exemple, réduire sa taille; ce qui équivaut à perdre toutes les données qu'elle contient...

Dans tous les cas, la prudence recommande de sauvegarder vos données avant d'utiliser **fdisk**.

12.2. fdisk pour un système

```
[root@tchana ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0004df76

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    41943039    19921920   8e  Linux LVM

Disk /dev/mapper/VolGroup-lv_root: 18.2 GB, 18249416704 bytes, 35643392 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/VolGroup-lv_swap: 2147 MB, 2147483648 bytes, 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

[root@tchana ~]#
```

`fdisk` vs type de disque d'installation du système **Proxmox Backup Server**.



On affiche tous les disques et leurs partitions.

```
[root@pbs-1 ~]# cat /proc/partitions

major minor  #blocks  name

   8        0   33554432 sda
   8        1       1007 sda1
   8        2     524288 sda2
   8        3   24640512 sda3
   8       16  419430400 sdb
  11        0    1234460 sr0
[root@pbs-1 ~]#
```

Le premier disque est celui sur lequel est installé Proxmox Backup Server.

VIII- Quincaillerie TCP/IP

1. Matériel

Le périphérique le plus courant sur PC pour construire un réseau local est la carte éthernet. Il en existe de nombreux modèles, de plusieurs fabricants, mais elles répondent toutes à la même norme **IEEE 802.3** et sont (*presque*) toutes capables de communiquer entre elles.

Éthernet est un réseau "démocratique" dans le sens où il n'y a pas de machine centrale: la gestion du réseau est partagée à égalité entre tous les ordinateurs qui y sont connectés. Ce principe présente des avantages et des inconvénients, mais dans l'ensemble il a fait ses preuves.

2. Connexion interne

Physiquement, une carte éthernet se présente comme une carte d'extension standard, à insérer dans une fente libre du PC. Il existe des modèles pour les bus **ISA** 8 bits, **ISA** 16 bits, et **PCI**. Les cartes communiquent avec la machine par une ligne d'**IRQ**, une adresse de base (***ioport***) et éventuellement un canal **DMA**. Ces 3 caractéristiques peuvent être fixées par le constructeur ou configurables par l'utilisateur. Dans ce dernier cas, la configuration peut se faire soit par logiciel, soit par la manipulation de "cavaliers" (*jumpers*) sur la carte.

2.1. 10 Base-T

Cette norme utilise de la paire torsadée, comme le fil de téléphone. Le connecteur (***RJ45***) est aussi semblable aux connecteurs téléphoniques modernes.

Les différentes cartes éthernet sont toutes reliées à un boîtier électronique central, appelé **répartiteur** (*hub en anglais*). C'est ce répartiteur qui s'occupe de commuter les signaux entre les différents câbles qui lui arrivent. En fait, si le réseau est gros, les ordinateurs ne sont pas tous reliés au même répartiteur. Chaque répartiteur accepte un certain nombre de connexions directes (*typiquement, entre 4 et 16*), mais peut être en plus relié lui-même à d'autres répartiteurs.

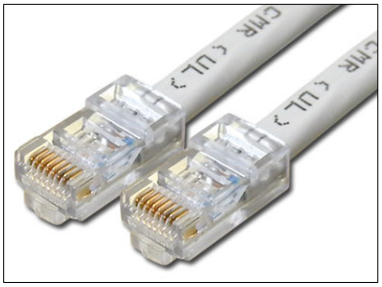


Figure 11: Câble RJ45.

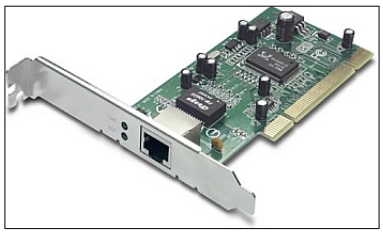


Figure 12: Carte réseau.

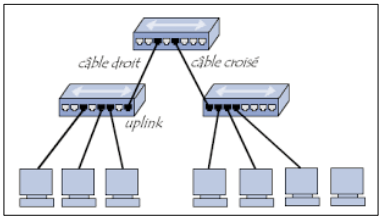


Figure 13: Répartiteur.

3. Les adresses éthernet

Chaque carte éthernet est identifiée par une adresse unique, fixée par le constructeur et inscrite "en dur" sur la carte, appelée **adresse MAC**[24].

Cette adresse a une taille de 6 octets, généralement donnée sous forme hexadécimale, les octets étant séparés par des doubles points. Par exemple, **00:00:C0:CE:7F:2E** est l'adresse d'une carte éthernet, de marque Western Digital, modèle WD8003. Les **trois premiers** octets identifient le fabricant, les **trois derniers** identifient le modèle et le numéro "de série".

Si votre carte éthernet est détectée au démarrage de Linux (*ce qui est généralement le cas*), vous pourrez voir cette adresse dans les messages de démarrage. Voici par exemple un extrait des messages de démarrage d'un PC (*obtenu par la commande **dmesg***):

```
[...]
PPP line discipline registered.
wd.c:v1.10 9/23/94 Donald Becker (becker@cesdis.gsfc.nasa.gov)
eth0: WD80x3 at 0x300, 00 00 C0 CE 7F 2E WD8003, IRQ 5, shared memory at 0xd8000-0xd9fff.
Partition check:
[...]
```

eth0 désigne la première interface éthernet, correspondant à la première carte reconnue. WD8003 est le nom de modèle, les autres informations sont des indications sur l'interface de communication avec le microprocesseur utilisé.

L'adresse matérielle permet au protocole éthernet de délivrer un message à son destinataire. Ce dernier connaît de la même façon l'**adresse MAC** de l'expéditeur.

Nous ne nous étendrons pas plus sur le protocole éthernet, qui dépasse largement le cadre de ce document.

4. Configuration d'une carte éthernet

La configuration d'une carte éthernet est généralement simple. Soit la carte est détectée au démarrage, (*dans le noyau ou en module*) et vous n'avez rien à faire, soit elle n'est pas supportée par Linux (*très rare*) et vous n'avez plus qu'à changer de carte!

En général, les distributions modernes (***Debian***, ***RedHat*** *et dérivées,* ***SuSE****...*) reconnaissent les cartes éthernet les plus courantes à l'installation et installent le noyau ou les modules nécessaires.

24 **adresse MAC**: (*rien à voir avec les MacIntosh d'Apple*) ou encore adresse physique, adresse matérielle (*hardware*).

IX- Introduction aux réseaux IP

À consulter en cas de problèmes de communication seulement

1. Référence

http://www.linux-france.org/article/formation/net.html par Vincent Defert - 28 Nov. 1998.

2. Introduction

Un réseau est un ensemble de dispositifs matériels et logiciels permettant à 2 machines ou plus de communiquer. Pour mettre en évidence les concepts importants, on peut utiliser l'analogie avec la vie quotidienne.

Imaginez que vous (*M. Dupond*) ayez un client (*M. Schmidt*) au téléphone. Celui-ci vous soumet un problème délicat. Vous attirez l'attention de votre collègue (*M. Jones*) et lui griffonnez quelques mots sur un bout de papier. M. Jones réfléchit un instant et vous griffonne sa réponse.

MM. Dupond, Schmidt et Jones représentent les machines connectées. Ces personnes sont reliées par 2 réseaux ("Liaison téléphonique" et "Liaison visuelle") auxquels ils sont reliés par 2 types d'interfaces (*"Combiné téléphonique" et "Papier + Crayon"*). M. Dupond, qui possède les 2 types d'interfaces, permet d'établir une communication entre M. Schmidt et M. Jones: il sert de **passerelle** entre les 2 réseaux. Enfin, chaque personne parle pendant un temps puis attend une réponse de son interlocuteur. Ils communiquent en échangeant des paquets d'informations.

A la lumière de cet exemple, on peut donc formuler ainsi les définitions des mots importants:

Réseau

Support permettant les échanges. Ce terme générique peut aussi bien désigner un câble reliant 2 machines qu'une ligne spécialisée (*dont l'autre extrémité est reliée à une installation complexe, mais qui est vue comme un support ordinaire du point de vue d'une interface*).

Le terme de réseau désigne aussi la totalité de l'installation (*machines, interfaces et câblage*). Le contexte permet de distinguer facilement le sens dans lequel il doit être compris.

Interface

Dispositif assurant la connexion de la machine à un réseau. Les interfaces utilisées avec les réseaux informatiques sont les cartes réseaux, les modems et les ports parallèles.

Routeur

Machine assurant l'interconnexion de plusieurs réseaux. On parle aussi de passerelle (*en anglais: **"router"** et **"gateway"***). Il peut s'agit d'une machine du réseau dans laquelle on a installé plusieurs interfaces ou d'un appareil spécialisé n'assurant que cette fonction.

Paquet

Unité de transport d'information.

Protocole

Ensemble de règles régissant les échanges d'informations.

Adresse

Identification des éléments intervenant dans la communication (*interfaces et réseaux*). L'analogie postale est particulièrement bonne: le nom de la rue correspond à l'adresse de réseau et le numéro de la maison à l'adresse de l'interface.

3. Réseaux IP

Les réseaux **IP** sont caractérisés par leur indépendance par rapport au matériel et par la possibilité d'établir une communication entre 2 machines situées sur des réseaux différents (*on dit que c'est un protocole "routable"*). Ces avantages découlent du mode d'adressage choisi. On affecte à chaque interface une adresse logique découlant de l'adresse du réseau auquel elle est connectée.

On fait ainsi d'une pierre 2 coups. Puisqu'il s'agit d'une adresse logique, on peut facilement faire évoluer le matériel (*il suffit de donner à la nouvelle machine l'adresse de l'ancienne*) et puisqu'elle découle de l'adresse du réseau auquel elle est reliée, il est facile de détecter si la machine à laquelle on veut envoyer des données est sur le même réseau ou non (*auquel cas on doit passer par un routeur*).

Prenons un exemple simple:

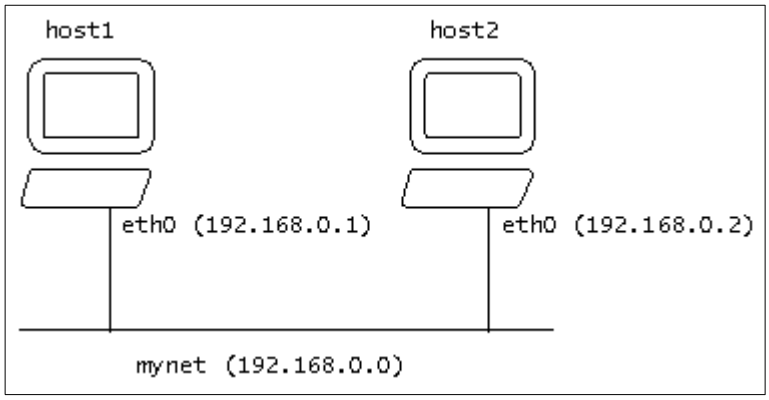


Figure 14: Exemple de réseau.

4. Réseau local isolé

host1 et **host2** sont les noms des machines, **eth0** le nom des interfaces, et **mynet** le nom du réseau (*on peut aussi l'appeler "mon petit réseau local"; ça n'a aucune importance si ce n'est faciliter les conversations entre humains*).

On a donc commencé par choisir une adresse de réseau: **192.168.0.0** (*nous verrons plus loin pourquoi*). Ensuite, on affecte une adresse logique à chacune des interfaces qui y est reliée, l'adresse de l'interface découlant de celle du réseau: **192.168.0.1**, **192.168.0.2**, etc.

Les adresses **IP** sont des nombres codés sur 32 bits. Par commodité, on les représente en décimal sous la forme de 4 nombres (*de 0 à 255 chacun*) séparés par des points (*"dotted quad" en anglais*).

Nous avons vu que l'adresse d'une interface découle de celle du réseau. La relation entre les 2 consiste à "couper" l'adresse en 2 parties. Les bits situés à gauche du point de coupure constituent la partie fixe (*l'adresse du réseau*) et ceux situés à droite sont disponibles pour numéroter les interfaces. L'adresse du réseau a donc tous les bits de la partie variable à 0. L'adresse dont tous les bits de la partie variable sont à 1 est réservée.

D'un point de vue arithmétique, on peut donc considérer que l'adresse de réseau peut être déterminée à partir de l'adresse d'une interface en faisant un **ET** logique entre l'adresse de l'interface et un nombre dont tous les bits de la partie fixe sont à 1 et ceux de la partie variable sont à zéro. Ce nombre est appelé **masque de réseau** (***netmask*** *en anglais*). Le **netmask** est représenté sous forme de dotted quad, comme les adresses.

Selon la valeur de l'adresse **IP**, on distingue plusieurs classes de réseaux (*correspondant à la position du point de coupure*). Ces classes sont simplement des conventions destinées à faciliter l'attribution des plages d'adresses.

- **Classe A**: le bit de poids le plus fort est à 0. Le netmask correspondant est **255.0.0.0**. Les réseaux **0.0.0.0** et **127.0.0.0** étant réservés, la classe A correspond donc à 126 réseaux de 16 millions de machines.
- **Classe B**: le bit de poids le plus fort est à 1, le suivant à 0. Son netmask est **255.255.0.0**. Elle comporte donc environ 16000 réseaux de plus de 65534 machines.
- **Classe C**: les 2 bits de poids le plus fort sont à 1, le suivant à 0. Le netmask est **255.255.255.0** et on a 2 millions de réseaux de 254 machines.

Il existe d'autres classes spéciales ou expérimentales sur lesquelles nous ne nous étendrons pas. Enfin, il est possible de créer des sous-réseaux à partir d'une plage d'adresses donnée en utilisant un netmask comportant plus de bits à 1 que le netmask conventionnel. Nous reviendrons sur ce point par la suite.

Remarque: Pour indiquer une adresse de réseau et son netmask, on utilise souvent une notation consistant à faire suivre l'adresse d'un **slash** (*"/"*) et du nombre de bits à un dans le netmask. Exemples:

- Adresse **192.168.12.0** - netmask **255.255.255.0** => **192.168.12.0/24**
- Adresse **10.0.0.0** - netmask **255.0.0.0** => **10.0.0.0/8**
- Adresse **10.0.0.0** - netmask **255.192.0.0** => **10.0.0.0/10**
- Adresse **172.16.128.0** - netmask **255.255.128.0** => **172.16.128.0/17**

Dans le cas d'un réseau local privé, on peut choisir n'importe quelle adresse. Dans le cas d'un réseau local connecté à Internet, votre fournisseur de connectivité **IP** vous indique la plage d'adresses qui vous a été allouée.

Certains numéros de réseaux ont été réservés pour les réseaux locaux privés (***RFC 1597***):

- **Classe A**: **10.0.0.0** (*1 réseau*)
- **Classe B**: **172.16.0.0** à **172.31.0.0** (*16 réseaux*)
- **Classe C**: **192.168.0.0** à **192.168.255.0** (*256 réseaux*)

Pour en finir avec les adresses réservées, signalons encore 2 cas: l'adresse de **loopback** et les adresses de **broadcast**.

L'adresse de **loopback** (***127.0.0.1***) correspond à une interface fictive présente sur toutes les machines, l'interface de loopback (***lo*** *ou* ***lo0***). Elle permet à la machine de s'envoyer à elle-même des paquets. Ça a peut être l'air bête, mais c'est très utile. On peut ainsi utiliser des applications réseau sans disposer d'interface physique ou sans qu'elle soit reliée au réseau. On peut par exemple, installer les programmes client et serveur d'une application dans la même machine pour aller faire une démonstration chez un (*futur*) client ou encore travailler chez-soi à la conception d'un site Internet, etc.

L'adresse de **broadcast** associée à un réseau correspond à une partie machine dont tous les bits sont à 1. Elle sert en particulier à déterminer quelle adresse physique correspond à une adresse **IP** donnée. Une requête envoyée à l'adresse de **broadcast** est reçue par toutes les machines, mais seule celle dont l'interface correspond à l'adresse demandée répond.

La question de l'adressage ayant été suffisamment traitée pour le moment, reste à aborder celle de la communication. Communiquer, c'est échanger des informations. Les règles régissant ces échanges sont appelées protocoles. Les principaux protocoles rencontrés avec les réseaux **IP** sont les suivants (*liste loin d'être exhaustive*).

Nom	Description
IP (*Internet Protocol*)	Échange de paquets entre nœuds
ICMP (*Internet Control Message Protocol*)	Transmission de messages d'erreur et de contrôle entre machines et passerelles
ARP (*Address Resolution Protocol*)	Correspondance entre adresse IP et adresse physique
RARP (*Reverse Address Resolution Protocol*)	Correspondance entre adresse physique et adresse IP
TCP (*Transmission Control Protocol*)	Transmission de données en mode connecté *(mode "Stream")*
UDP (*User Datagram Protocol*)	Transmission de données sans connexion (*mode "Datagram"*)
FTP (*File Transfer Protocol*)	Services de haut niveau de transfert de fichiers
HTTP (*HyperText Transfer Protocol*)	Serveurs Web
NNTP (*Network News Transfer Protocol*)	Serveurs de News
SMTP (*Simple Mail Transfer Protocol*)	Transmission de courrier électronique
POP (*Post Office Protocol*)	Gestion de boîtes aux lettres électroniques
Telnet	Échange de terminal

Cette liste permet de voir que les protocoles s'adressent à différents niveaux. Un programme de transfert de fichiers fournira une interface utilisateur au protocole **FTP** qui s'appuie lui même sur **TCP**, lequel utilise **IP**, ce dernier exploitant **ARP** et **RARP**. C'est pour cette raison qu'on parle souvent de la "pile **TCP/IP**" ou de la "suite de protocoles **TCP/IP**".

Les spécifications de ces protocoles sont appelées **RFC** (*Request For Comments*). On peut les trouver sur tous les grands sites **FTP** notamment: les sites universitaires, ceux des organismes de recherche et ceux des sociétés liées à l'internet. Exemples: ftp.lip6.fr, ftp.inria.fr, ftp.nic.fr.

Notez au passage que le terme "**TCP/IP**" est utilisé pour désigner des choses qui ne mettent pas nécessairement en œuvre le protocole **TCP**: "**TCP/IP**" est devenu un nom propre indépendant des abréviations qui le composent.

Supposons maintenant qu'une machine soit utilisée pour faire plusieurs tâches (*situation normale*): serveurs Web, **FTP**, **SMTP** et **POP**. Cette machine n'a qu'une interface, donc qu'une adresse **IP**. Le problème qui se pose est de trouver un moyen d'indiquer à cette machine le programme auquel transmettre le paquet qu'on lui envoie.

La solution à ce problème est la notion de port. On affecte un numéro à chacun des services offerts par la machine (*80 pour le serveur **HTTP**, 21 pour le serveur **FTP**, 25 pour le serveur **SMTP** et 110 pour le serveur **POP***).

Notez qu'on peut transmettre des données à un port selon 2 modes: connecté (***TCP***) ou non (***UDP***). Le mode connecté (*ou mode Stream*) garantit que les données seront reçues en totalité et dans l'ordre où elles ont été envoyées. Le mode non-connecté (*ou mode Datagram*) permet d'envoyer un message isolé sans garantie qu'il soit reçu.

Cette dernière définition peut surprendre, mais ce mode est très utile au cours d'un transfert de fichier (*donc mode Stream*); par exemple, on peut avoir besoin de transmettre un message d'alerte. On enverra alors cette information sous la forme d'un datagramme sur le même port (*puisqu'il est destiné à la même application*). Le programme de réception peut détecter l'arrivée de ces données "hors bande" et appeler la procédure adéquate sans interférer avec le transfert en cours (*sauf s'il s'agit d'une demande d'interruption...*). Si par malchance le message d'erreur n'était pas reçu, il pourrait être envoyé une seconde fois. L'absence de garantie d'acheminement n'est donc pas un handicap.

5. Le routage

Après avoir passé en revue les notions de base sur un exemple simple (*réseau local isolé*), nous sommes armés pour passer au cas de 2 réseaux locaux interconnectés. Cette situation se généralise dans le cas d'un réseau relié à l'Internet, c'est à dire à un nombre impressionnant d'autres réseaux. Nous parlerons aussi des connexions en "dial-up" (*par modem*).

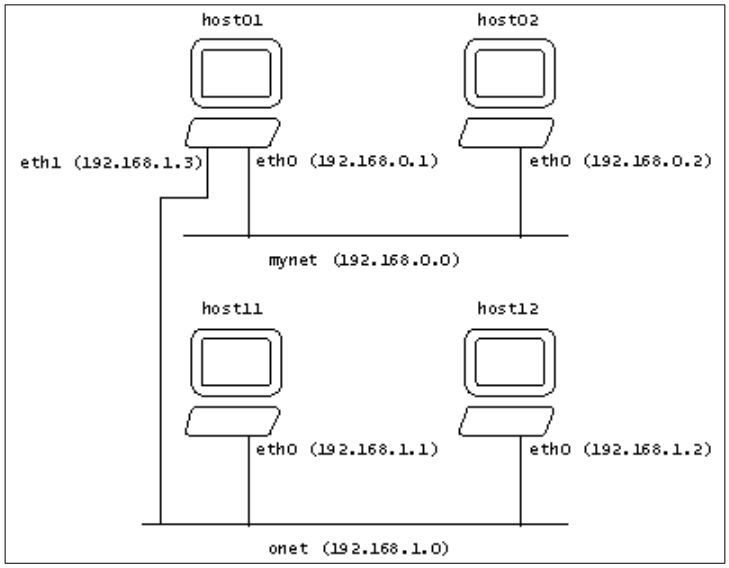


Figure 15: Exemple de routage.

6. Réseaux locaux interconnectés

Pour permettre à 2 machines situées sur des réseaux différents de communiquer entre-elles, il est nécessaire d'établir une liaison entre ces 2 réseaux. La machine "**host01**" remplit cette fonction de **passerelle** entre les 2 réseaux; elle dispose de 2 interfaces reliées chacune à un réseau. Il ne reste qu'à indiquer à la machine comment utiliser ces interfaces. C'est le rôle de la table de routage.

Chaque paquet émis possède une adresse de départ et une adresse d'arrivée. Lorsqu'une application envoie un paquet, la fonction système responsable de son acheminement parcoure la table de routage de la machine pour déterminer quelle interface elle devra utiliser.

Voici sous forme simplifiée les tables de routage des machines **host01**, **host02** et **host11**.

Machine host01				Machine host02			
Adresse	**Masque**	**Interface**	**Passerelle**	**Adresse**	**Masque**	**Interface**	**Passerelle**
127.0.0.0	255.0.0.0	lo0	néant	127.0.0.0	255.0.0.0	lo0	néant
192.168.0.0	255.255.255.0	eth0	néant	192.168.0.0	255.255.255.0	eth0	néant
192.168.1.0	255.255.255.0	eth1	néant	0.0.0.0	0.0.0.0	eth0	192.168.0.1

Machine host11			
Adresse	**Masque**	**Interface**	**Passerelle**
127.0.0.0	255.0.0.0	lo0	néant
192.168.1.0	255.255.255.0	eth0	néant
0.0.0.0	0.0.0.0	eth0	192.168.1.3

La ligne des tables de routage de **host02** et **host11** dont l'adresse de destination est **0.0.0.0** correspond à la route par défaut, c'est à dire à l'interface à utiliser quand aucune des autres entrées de la table ne convient.

Supposons maintenant que **host02** veuille envoyer un paquet à **host11**. L'adresse d'arrivée est donc **192.168.1.1**. **host02** parcoure chaque entrée de sa table de routage et vérifie si l'adresse d'arrivée remplit les conditions indiquées.

Sur la première ligne, il applique le masque **255.0.0.0** à **192.168.1.1** et trouve donc **192.0.0.0**, alors que cette ligne est donnée pour **127.0.0.0**. On passe donc à la suivante: **192.168.1.1** et **255.255.255.0** donne **192.168.1.0**, ce qui ne correspond pas au **192.168.0.0** attendu. Il ne reste plus que la route par défaut: le paquet sera donc transmis à **host01** (**192.168.0.1**).

Lorsque **host01** reçoit ce paquet, il applique la même technique. Sur la première ligne, le résultat est identique au cas précédent. Sur la 2ème ligne, **192.168.1.1** et **255.255.255.0** donne **192.168.1.0**, ce qui ne correspond pas au **192.168.0.0** attendu. Par contre, la condition de la 3ème ligne est satisfaite: le paquet sera donc transmis sur **eth1** et reçu par **host11**.

Le routage ressemble à ces jeux éducatifs composés d'un ensemble de formes en plastique et d'une boîte comportant des trous appropriés. L'enfant doit essayer les différents trous jusqu'à ce qu'il trouve celui qui convient à la forme qu'il veut ranger dans la boîte. Moralité: **TCP/IP**, c'est enfantin!

7. Poste avec connexion dial-up

Le cas d'une connexion "dial-up" est intéressant parce qu'il met en œuvre la modification dynamique de la table de routage. Lorsque le modem établit la connexion avec son homologue à l'autre bout de la ligne téléphonique, une nouvelle interface (***ppp0***) est créée et 2 entrées sont ajoutées à la table de routage (*une vers le modem distant et une route par défaut*).

Les différentes étapes sont les suivantes:

- Le modem local numérote, le modem distant décroche et tous 2 négocient le protocole de communication (*vitesse, compression, etc*).
- Une fois le lien physique établi, les machines négocient les paramètres du lien **PPP**. C'est à ce moment que l'interface associée au modem local est créée. La machine distante lui assigne alors une adresse **IP** et indique également l'adresse de la passerelle pour la route par défaut.
- Lors de la déconnexion, les entrées ajoutées dans la table de routage sont supprimées, de même que l'interface **ppp0**.

Supposons par exemple que le modem local reçoive l'adresse **192.168.1.1** et que celle du modem distant soit **192.168.1.2**, on aura alors les tables de routage suivantes:

7.1. Machine host1 non connectée

Adresse	*Masque*	*Interface*	*Passerelle*
127.0.0.0	255.0.0.0	lo0	néant

7.2. Machine host1 connectée

Adresse	*Masque*	*Interface*	*Passerelle*
127.0.0.0	255.0.0.0	lo0	néant
192.168.1.0	255.255.255.0	ppp0	néant
0.0.0.0	0.0.0.0	ppp0	192.168.1.2

8. Domain Name System

Jusqu'à maintenant, nous ne nous sommes occupés que des adresses des machines, mais dans la réalité on utilise leurs noms. Pour la plupart des gens, **www.yahoo.fr** est beaucoup plus facile à retenir que **195.67.49.44**.

Toute machine peut utiliser un fichier local contenant la correspondance entre adresses **IP** et noms de machines. Le problème avec cette approche est qu'elle n'est valable que pour un réseau local qui ne change pas très souvent (*car il faut alors mettre à jour les tables de toutes les machines!*).

L'objet du **Domain Name System** est de fournir un moyen pour transformer les noms de machines en adresses **IP** à l'échelle planétaire. Le principe consiste à organiser les noms des machines de façon hiérarchique.

Ainsi, le nom **www.yahoo.fr** comporte 3 niveaux, lus de droite à gauche. Chaque niveau gère la totalité des niveaux immédiatement inférieurs. Un certain nombre de serveurs gèrent les domaines de premier niveau (*ici: **fr***), ceux-ci déléguant à leur tour la gestion des niveaux inférieurs.

En résumé, les serveurs "racine" gèrent les domaines de plus haut niveau (*com, fr, uk, etc*), les **Network Information Centers** nationaux gérant tous les niveaux inférieurs (*ex. le NIC France gère ibp.fr, gouv.fr, yahoo.fr, etc*). Yahoo! France gère tous les noms se terminant par **yahoo.fr**, et ainsi de suite.

Lorsqu'une machine veut transformer un nom en adresse **IP**, elle fait appel à un serveur de noms qui se chargera d'exploiter toute l'arborescence de ses confrères pour obtenir le renseignement. Les serveurs de noms utilisent un "cache" pour éviter de déclencher une recherche à chaque demande.

Le système de délégation permet de résoudre le problème des mises à jour (*à la durée de vie des informations copiées dans les caches près*) et de laisser une totale liberté de gestion à chaque niveau.

Chaque serveur en charge d'un domaine doit être secondé par au moins un autre serveur. Ce sont les **DNS secondaires**, qui recopient à intervalles réguliers les données du serveur primaire et peuvent traiter les demandes en cas d'indisponibilité du serveur primaire.

X- L'éditeur vi

1. Référence

http://www.iro.umontreal.ca/~dift3830/vi.html. *(Dernière consultation, le 30 mai 2016. et le 7 novembre 2018, ce lien n'est toujours plus fonctionnel.)*

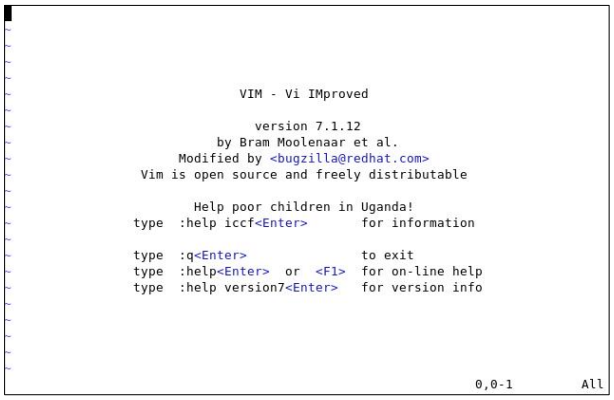
vi est un éditeur de texte très puissant. Sa convivialité par contre lui fait défaut. Ceci dit, il est toujours utile d'en connaître les rudiments, car son omniprésence est presque garantie sur les systèmes modernes.

La documentation de **vi** étant très abondante, on se limitera pour cette démo aux commandes les plus usuelles.

Tout d'abord l'invocation; on peut invoquer **vi** à partir du **shell** de plusieurs façons dont voici quelques-unes:

- **vi**: ouvre **vi** avec un contenu vide.
- **vi nom_de_fichier**: ouvre un fichier et l'affiche à l'écran.
- **vi +nom_de_fichier**: ouvre un fichier et positionne le curseur à la fin de celui-ci.

Dès son invocation, **vi** se met en **mode commande**, dans ce mode il est possible d'entrer les commandes qui seront vues plus bas. Si on tape une commande susceptible de modifier un texte *(insertion d'un caractère par exemple)*, **vi** bascule en **mode édition**; dans ce mode tout caractère entré sera considéré comme faisant partie du texte, tandis que les caractères saisis en **mode commande**, seront eux, interprétés comme étant des commandes et ne seront jamais rajoutés au texte.



Afin de basculer du **mode édition** au **mode commande** il suffit de presser la touche [**Échap**].

Nous allons commencer par invoquer **vi** à partir du **shell** en tapant:

```
vi
```

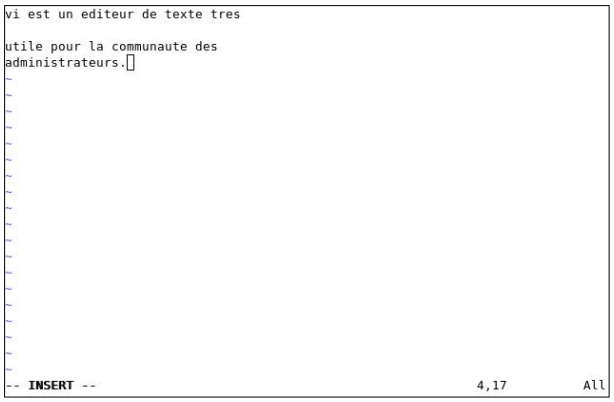
Ce qui devrait donner l'affichage ci-contre:

vi est déjà en **mode commande**, pour le faire passer en **mode édition**, on tape la commande **i** *(insert)* qui nous permettra d'**i**nsérer du texte.

Après avoir entré le texte suivant:

```
vi est un éditeur de texte très

utile pour la communauté des
administrateurs.
```

On obtiendra l'affichage ci-contre.



Après cela, on pourrait passer en **mode commande** par simple pression sur la touche [**Échap**].

Une fois en **mode commande**, on voudrait par exemple, éliminer la ligne blanche qui se trouve juste après la première. Pour cela on positionne le curseur a la hauteur de la 2e ligne et on entre **dd**.

Ceci aura pour effet de supprimer la ligne.

Les commandes abondent dans **vi**, c'est pour cette raison qu'on n'en citera que quelques-unes.

Si on est satisfait, il ne nous reste plus qu'à sauvegarder le document sous le nom **texte1.txt** à l'aide de la commande suivante:

```
:w texte1.txt
```

(Pour les sauvegardes ultérieures, il n'est pas nécessaire d'ajouter le nom de fichier).

Afin de quitter **vi** il suffit de taper la commande:

```
:q texte1.txt
```

Commande	**Effets**
i (insert)	Insère un texte au curseur
I	Insère au début de la ligne
a (append)	Insère après le curseur
A	Insère à la fin de la ligne
Les flèches	Pour les déplacements
Ctrl-F (forward)	Défile d'un écran vers le bas
Ctrl-B (backward)	Défile d'un écran vers le haut
nG (goto)	Va à la nième ligne dans le texte
G	Va à la fin du texte
x	Efface le caractère courant
dd	Efface la ligne courante
D	Efface depuis la position du curseur jusqu'à la fin de la ligne
db (DeleteBegining)	Efface depuis la position courante jusqu'au début de la ligne
/chaîne	Recherche la chaîne 'chaîne' dans le texte, on peut taper 'n' *(next)* pour voir l'occurrence suivante
:w fichier	Copie le texte courant sur le disque sous le nom fichier
:wq (write & quit)	Écrit le fichier sur le disque et quitte vi.
:q!	Quitte sans sauvegarder.
:set nu	Affiche le numérotage des lignes.

Victoire totale, hissons la bannière de la victoire.

XI- Projet ISPconfig

1. Projet ISPconfig

Le but final du projet ISPconfig est de créer son propre serveur ISPconfig qui peut héberger un service de messagerie électronique, un ou plusieurs sites Web et même un site de commerce en ligne.

- Le premier pas est la création d'un serveur Proxmox VE pour faciliter le développement de ce projet.
- Vient ensuite la création d'une machine virtuelle pour l'installation d'un serveur minimal Debian.
- L'étape suivante est l'installation et la configuration du serveur ISPconfig roulant sous le serveur Debian.
- On ajoute un pare-feu UCG Ultra pour protéger le réseau.
- On termine en branchant le tout directement à l'Internet.

2. Proxmox

Habituellement, l'installation de PVE s'effectue sur une machine qui lui est complètement dédiée. Vu que nous voulons seulement nous familiariser avec le comportement de cet environnement, nous allons commencer par installer Proxmox VE sur une machine virtuelle sous un véritable serveur Proxmox VE *(Nested Virtualization)*.

Une fois familiarisé avec l'installation, vous pourrez répéter les mêmes procédures sur une machine physique dédiée à PVE.

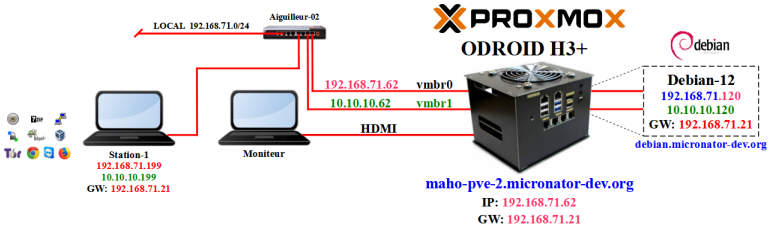
Vous pourrez vérifier toute modification à l'environnement PVE sur la machine virtuelle avant de l'implémenter sur la machine physique.

2.1. Cahier

Cahier-01: Proxmox VE virtuel *Cahier-02:* Proxmox VE physique

3. Debian minimal

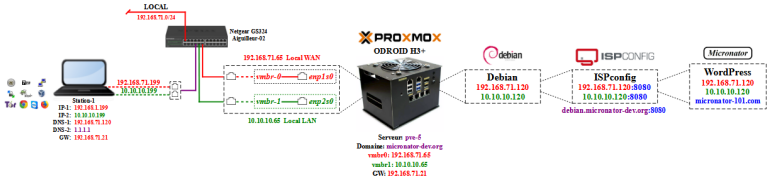
Installation d'un serveur **Debian** minimal qui servira d'hôte pour héberger un serveur **ISPconfig**.



3.1. Cahier

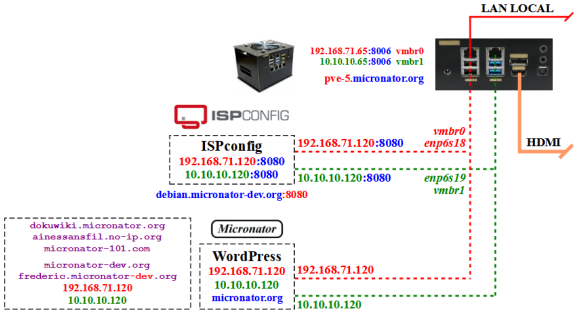
Cahier-01: Debian minimal

4. ISPconfig



Avec plus de 40,000 téléchargements par mois, ISPconfig est un système à étudier et à mettre à l'essai.

Ce didacticiel vous guidera tout au long de l'installation de votre propre serveur ISPConfig-3.2 à l'aide du programme d'auto-installation qui suit les anciens guides "Perfect Server" mais qui est plus modulaire et facile à suivre.



Ce guide fonctionne pour Debian 10 à Debian 12, Ubuntu 20.04 et Ubuntu 22.04. Il ne prend actuellement en charge que l'architecture CPU x86_64 *(également connue sous le nom d'AMD64)*, tandis qu'ARM n'est pas pris en charge. Le guide nécessite un système d'exploitation de base fraîchement installé et vide, n'essayez pas de l'utiliser sur un système sur lequel vous avez déjà configuré d'autres services.

Le panneau de contrôle d'hébergement Web d'ISPConfig-3 vous permet de configurer les services suivants via un navigateur Web: serveur Web Apache ou nginx, serveur de messagerie Postfix, serveur Dovecot IMAP/POP3, MySQL, serveur de noms BIND, PureFTPd, Rspamd ou Amavis, ClamAV, et bien d'autres.

Cette configuration couvre Apache *(au lieu de nginx)*, BIND et Dovecot avec le balayeur de pourriels Rspamd.

Ce document contient une marche à suivre pour cloner un site WordPress vers ISPconfig de même que les répertoires des courriels des usagers.

4.1. Cahier

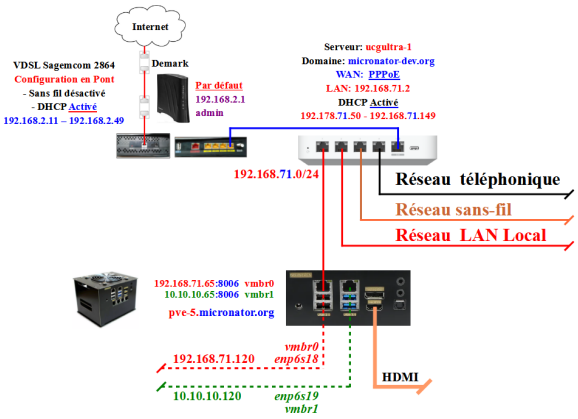
Cahier-01: ISPconfig

5. Pare-feu UCG Ultra

Référence: **https://infologo.ch/blog/quest-ce-que-unifi-ubiquiti/**.

UniFi est la gamme d'équipements réseau d'Ubiquiti comprenant différents modèles de points d'accès sans-fil, de routeurs, de commutateurs, de caméras de sécurité, d'appareils de contrôle, de téléphones VoIP et de produits de contrôle d'accès. L'équipement UniFi se positionne entre le matériel réseau d'entreprise et le matériel réseau domestique bon marché. Il offre plus de flexibilité et de fonctionnalités que la plupart des marques grand public, tout en étant moins coûteux et moins complexe que les solutions d'entreprise.

La gamme UniFi repose sur un gestionnaire qui peut-être installé sur différents ordinateurs ou cellulaires.



Tout votre système réseau est donc en local chez vous. Vous pouvez très bien y avoir accès depuis l'extérieur en passant par un nom de domaine.

Référence: **https://www.wifi-france.com/ubiquiti/ucg-ultra**.

Le routeur **Cloud Gateway Ultra** d'Ubiquiti est un routeur multi-WAN puissant et compact avec un ensemble complet de fonctionnalités de routage et de sécurité avancées. Il est idéal pour les petites et moyennes entreprises qui ont besoin d'un routeur fiable et performant pour protéger leur réseau et garantir une connectivité optimale.

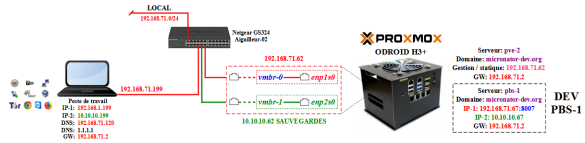
5.1. Cahier

Cahier-02: Pare-feu UCG Ultra

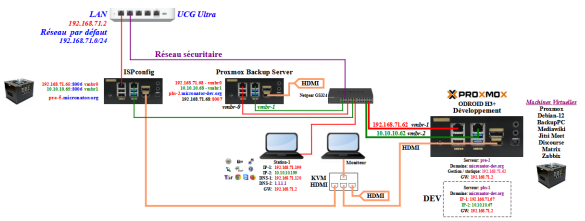
6. Proxmox Backup Server

Référence: **https://pbs.proxmox.com/docs/introduction.html#**.

Proxmox Backup Server est une solution de sauvegarde *client-serveur* de classe entreprise capable de sauvegarder des machines virtuelles, des conteneurs et des hôtes physiques. Elle est spécialement optimisée pour la plateforme Proxmox Virtual Environment et vous permet de sauvegarder vos données en toute sécurité, même entre des sites distants, en offrant une gestion facile via une interface Web.



Proxmox Backup Server virtuel



Proxmox Backup Server physique

Elle prend en charge la déduplication, la compression et le chiffrement authentifié *(AE)*. L'utilisation de `Rust` comme langage d'implémentation garantit des performances élevées, une faible utilisation des ressources et une base de code sûre et de haute qualité.

Proxmox Backup utilise un chiffrement de pointe pour la communication client-serveur et le contenu des sauvegardes. Toutes les communications client-serveur utilisent `TLS` et les données de sauvegarde peuvent être chiffrées côté client avant l'envoi, ce qui rend plus sûre la sauvegarde des données vers des cibles qui ne sont pas entièrement fiables.

Une fois familiarisé avec l'installation, vous pourrez répéter les mêmes procédures sur une machine physique dédiée à PBS.

Vous pourrez vérifier toute modification à l'environnement PBS sur la machine virtuelle avant de l'implémenter sur la machine physique.

6.1. Cahier

Cahier-01: Proxmox Backup Server virtuel

Cahier-02: Proxmox Backup Server physique

7. Linux-101

Pour connaître les rudiments de Linux ou rafraîchir vos connaissances, vous pouvez consulter ce document: *Cours Linux-101*. Ce cours donne un aperçu des fonctionnalités de base de Linux qui sont indispensables à toute personne qui désire se familiariser à l'environnement Linux et surtout comprendre et maîtriser les concepts de base.

Vous serez en mesure de recourir à la documentation en ligne (*man*), manipuler l'arborescence des fichiers, comprendre l'organisation générale du système, gérer les droits d'accès, découvrir les variables d'environnement, les fichiers particuliers, la quincaillerie réseau, utiliser les principales commandes bash, etc.

Un chapitre particulier, qu'il n'est pas nécessaire de maîtriser mais simplement connaître, explique les principes de base de la communication **TCP/IP**.

7.1. Cahier

Cahier-1: Linux-101

8. Utilitaires-101

Ce document, **Cahier-01** *Utilitaires* du cours "Utilitaires-101" décrit les logiciels prérequis: **DigestIT-2004**, **7-Zip**, **WinSCP**, **PuTTY**, **NotePad++**, **VirtualBox**, **Chrome**, **TOR** et *TeamViewer*. **AnyDesk** peut remplacer *TeamViewer*.

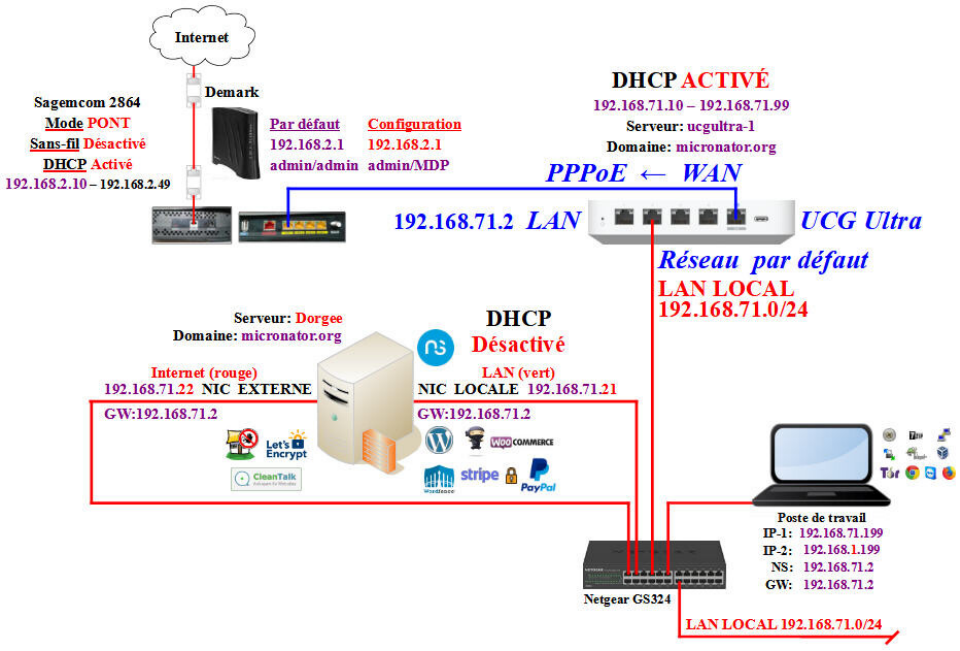
8.1. Cahier

Cahier-1: Utilitaires-101

XII- Cours NethServer

Le Serveur NethServer est en fin de vie depuis le 30 juin 2024.

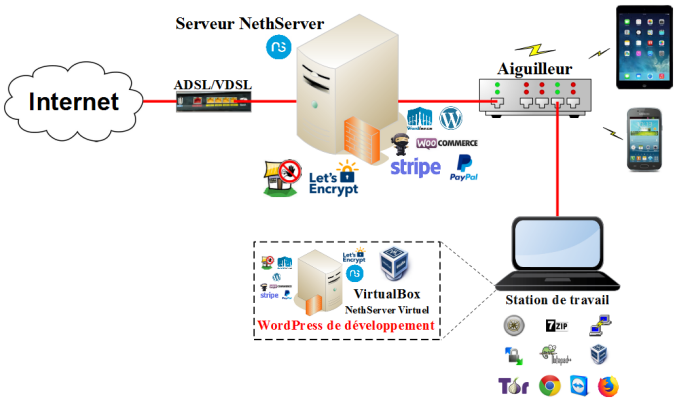
Soins palliatifs: Pare-feu UCG Ultra



1. Cours NethServer-101 - NethServer & Commerce en ligne

1.1. But final

Après avoir suivi le "Cours NethServer-101", vous posséderez un site de **Commerce en ligne**, fiable et hautement sécuritaire. De plus, vous pourrez utiliser un clone de votre site sur un *Serveur NethServer* virtuel, pour tester de nouvelles extensions et applications sans compromettre la sécurité ou l'intégrité de votre site en ligne.



1.2. Cahiers

Le Cours **NethServer-101**, se voulant une base solide pour la création d'un site de **Commerce en ligne**, il comprend plusieurs cahiers:

- *Cahier-01:* Les bases de Linux.
- *Cahier-02:* Installation et configuration des logiciels prérequis sur le poste de travail.
- *Cahier-03:* Création d'un *Serveur NethServer* virtuel.
- *Cahier-04: Serveur NethServer* LOCAL & Let's Encrypt.
- *Cahier-05:* Abonnement à un *FAI*, installation d'un modem *VDSL*, obtention d'un domaine *FQDN*[25] et installation d'un *Serveur NethServer* sur une quincaillerie physique.
- *Cahier-06:* Installation de *WordPress*.
- *Cahier-07:* Installation de l'extension de sécurité *Wordfence*.
- *Cahier-08:* Installation de l'extension de vente en ligne *WooCommerce* et création de comptes chez *Stripe* et *PayPal* pour les paiements en ligne.
- *Cahier-09:* Sauvegarde/restauration ou migration d'un site avec l'extension *Duplicator*.
- *Cahier-10:* Serveur mandataire inverse.
- *Cahier-11:* Sauvegarde/restauration avec *BackupPC*.
- *Cahier-12: WordPress & CleanTalk*.

1.3. Logiciels

Tous les logiciels nécessaires sont du domaine public ou LIBRE sous licence *GPL*; ils ne coûtent pas un sous. Le seul achat nécessaire est l'obtention d'un nom de domaine au prix initial de $15 CAD et son renouvellement annuel d'environ $30 CAD.

Cours NethServer-201 - NethServer & Applications

Cahier-01: Dolibarr
Cahier-02: Odoo-12
Cahier-03: MediaWiki
Cahier-04: DokuWiki
Cahier-05: Moodle
Cahier-06: Proxmox VE
Cahier-07: Flectra
Cahier-08: Self Service Password
Cahier-09: Forum NodeBB
Cahier-10: Forum Discourse
Cahier-11: diaspora*

Cours NethServer-301 - NethServer & Active Directory

Cahier-01: RSAT
Cahier-02: Migration LDAP vers Active Directory
Cahier-03: SSP & Active Directory
Cahier-04: Jonction de stations à AD

Cours NethServer-401 - Surveillance Zabbix

Cahier-01: Zabbix - Installation
Cahier-02: Zabbix - Alertes
Cahier-03: Zabbix - Agents
Cahier-04: Zabbix & Émulateur ELM327
Cahier-05: Zabbix & Mise à niveau

25 **FQDN**: Dans le *DNS*, un **F**ully **Q**ualified **D**omain **N**ame *(FQDN, ou nom de domaine complètement qualifié)* est un nom de domaine qui révèle la position absolue d'un nœud dans l'arborescence DNS en indiquant tous les domaines de niveau supérieur jusqu'à la racine. On parle également de domaine absolu, par opposition aux domaines relatifs. Par convention, le FQDN est ponctué par un point final.
Référence: **https://fr.wikipedia.org/wiki/Fully_qualified_domain_name**.

Cours NethServer-501 - Service d'assistance Zammad

Cahier-01: Zammad - Installation
Cahier-02: Zammad - Création de billets
Cahier-03: Zammad - Sauvegarde & restauration

Cours NethServer-601 - Communication d'équipe Mattermost

Cahier-01: Mattermost - Installation
Cahier-02: Mattermost - Mise à niveau
Cahier-03: Mattermost - Récupération après désastre

Cours NethServer-701 - Matrix-Synapse

Cahier-01: Matrix-Synapse
Jitsi Meet & Mattermost

Cours NethServer-801 - Jitsi Meet

Cahier-01: Jitsi Meet
Cahier-03: Jitsi Meet & Forum Discourse
Cahier-04: Jitsi Meet & Matrix-Synapse
Cahier-04: Jitsi Meet & Mattermost

2. Commentaires et suggestions

RF-232 apprécie grandement échanger avec ses internautes. Vos commentaires et suggestions sont indispensables à l'amélioration de la documentation et du site **micronator.org**.

N'hésitez pas à nous transmettre vos commentaires et à nous signaler tout problème d'ordre technique que vous avez rencontré ou n'arrivez pas à résoudre. Tous vos commentaires seront pris en considération et nous vous promettons une réponse dans les plus brefs délais.

Brancher les aînés,
encourager l'Informatique LIBRE
et la diffusion du savoir

3. Boutique Micronator

Nous sommes heureux de vous présenter notre boutique en ligne dans laquelle vous trouverez certains de nos produits qui ne sont pas disponibles sur notre site principal. Nous vous laissons le plaisir de la parcourir:
https://www.micronator.org/affaires/boutique/.

3.1. Communications sécuritaires chiffrées SSL/TLS

Les communications avec **Stripe** et **PayPal** sont effectuées au moyen d'un **certificat SSL/TLS de 2048** bits émis par l'Autorité de Certification **Let's Encrypt**.

Faites vos achats en toute confiance, remplissez votre panier et réglez votre commande avec la carte bancaire de votre choix: **MasterCard**, **Visa**, **Discover**, **American Express**, etc.

Stripe

Le montant de votre facture est envoyé directement à **Stripe** qui s'occupe de tout. Les données de votre carte ne sont pas utilisées sur notre site. Les paiements sont sécurisés par le système **Stripe**. **Cliquez ici** pour voir les étapes de paiements; celles-ci sont sécurisées par le système **Stripe**.

PayPal

Il n'est pas nécessaire d'ouvrir un compte **PayPal**. Vous pouvez choisir la carte bancaire que vous désirez utiliser. **Cliquez ici** pour voir les étapes de paiements; celles-ci sont sécurisées par le système **PayPal**.

4. Médias sociaux

Twitter: **https://twitter.com/TuteurW**.

Facebook: **https://www.facebook.com/micronator**.

Crédits

Auteur: **Michel-André Robillard CLP**

Remerciement: **Tous les contributeurs GNU/GPL**.

Intégré par: **Michel-André Robillard CLP**

Contact: **michelandre at micronator.org**

Répertoire de ce document: E:\000_DocPourRF232_general\RF-232_Linux-101\RF-232_Linux-101_2025-01-09_14h42.odt

Historique des modifications**:**

Version	*Date*	*Commentaire*	*Auteur*
1.0.0	2017-01-05	Début.	M.-A. Robillard
1.0.1	2017-02-22	Note pour le cours Micronator-101.	M.-A. Robillard
1.0.2	2018-03-10	Divisé le Cahier-5 en Cahier-5A et Cahier-5B.	M.-A. Robillard
2.0.0	2018-12-17	Mise à jour pour le Cours NethServer-101.	M.-A. Robillard
2.0.1	2019-03-26	Correction de coquilles.	M.-A. Robillard
2.0.2	2019-05-03	Correction mineures.	M.-A. Robillard
2.1.0	2019-07-09	- Correction mineures. - Ajout d'une note pour le fichier /etc/inittab. - Changement des captures d'écran pour l'éditeur vi.	M.-A. Robillard
3.0.0	2024-12-13	Mise à jour pour ISPconfig.	M.-A. Robillard
3.0.1	2025-01-08	- Correction des figures 7 et 8. - Nouvelle mise en page et vérification	M.-A. Robillard

Index

M

N

O

P

Q

R

S

T

U

V

W

LICENCE PUBLIQUE GÉNÉRALE GNU

Version 3, du 29 juin 2007.
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Chacun est autorisé à copier et distribuer des copies conformes de ce document de licence, mais toute modification en est proscrite.

Traduction française par Philippe Verdy <verdy_p (à) wanadoo (point) fr>, le 30 juin 2007 (dernière correction du 4 janvier 2011).
Avertissement important au sujet de cette traduction française.
Ceci est une traduction en français de la licence “GNU General Public License” (GPL). Cette traduction est fournie ici dans l’espoir qu’elle facilitera sa compréhension, mais elle ne constitue pas une traduction officielle ou approuvée d’un point de vue juridique.
La Free Software Foundation (FSF) ne publie pas cette traduction et ne l’a pas approuvée en tant que substitut valide au plan légal pour la licence authentique “GNU General Public Licence”. Cette traduction n’a pas encore été passée en revue attentivement par un juriste et donc le traducteur ne peut garantir avec certitude qu’elle représente avec exactitude la signification légale des termes de la licence authentique “GNU General Public License” publiée en anglais. Cette traduction n’établit donc également aucun des termes et conditions d’utilisation d’un logiciel sous licence GNU GPL — seul le texte original en anglais le fait. Si vous souhaitez être sûr que les activités que vous projetez seront autorisées par la GNU General Public License, veuillez vous référer à sa seule version anglaise authentique.
La FSF vous recommande fermement de ne pas utiliser cette traduction en tant que termes officiels pour vos propres programmes; veuillez plutôt utiliser la version anglaise authentique telle que publiée par la FSF. Si vous choisissez d’acheminer cette traduction en même temps qu’un Programme sous licence GNU GPL, cela ne vous dispense pas de l’obligation d’acheminer en même temps une copie de la licence authentique en anglais, et de conserver dans la traduction cet avertissement important en français et son équivalent en anglais ci-dessous.

Important Warning About This French Translation.
This is a translation of the GNU General Public License (GPL) into French. This translation is distributed in the hope that it will facilitate understanding, but it is not an official or legally approved translation.
The Free Software Foundation (FSF) is not the publisher of this translation and has not approved it as a legal substitute for the authentic GNU General Public License. The translation has not been reviewed carefully by lawyers, and therefore the translator cannot be sure that it exactly represents the legal meaning of the authentic GNU General Public License published in English. This translation does not legally state the terms and conditions of use of any Program licenced under GNU GPL — only the original English text of the GNU LGPL does that. If you wish to be sure whether your planned activities are permitted by the GNU General Public License, please refer to its sole authentic English version.
The FSF strongly urges you not to use this translation as the official distribution terms for your programs; instead, please use the authentic English version published by the FSF. If you choose to convey this translation along with a Program covered by the GPL Licence, this does not remove your obligation to convey at the same time a copy of the authentic GNU GPL License in English, and you must keep in this translation this important warning in English and its equivalent in French above.

Préambule
La Licence Publique Générale GNU (**“GNU General Public License”**) est une licence libre, en **“copyleft”**, destinée aux œuvres logicielles et d’autres types d’œuvres.
Les licences de la plupart des œuvres logicielles et autres œuvres de la pratique sont conçues pour vous ôter votre liberté de partager et modifier ces œuvres. À l’inverse, la Licence Publique Générale GNU a pour but de garantir votre liberté de partager et changer toutes les versions d’un programme — afin d’assurer qu’il restera libre pour tous les utilisateurs. Nous, la **Free Software Foundation**, utilisons la Licence Publique Générale GNU pour la plupart de nos logiciels; cela s’applique aussi à toute autre œuvre éditée de cette façon par ses auteurs. Vous pouvez, vous aussi, l’appliquer à vos propres programmes.
Quand nous parlons de logiciel libre (**“free”**), nous nous référons à la liberté (**“freedom”**), pas au prix. Nos Licences Publiques Générales sont conçues pour assurer que vous ayez la liberté de distribuer des copies de logiciel libre (et le facturer si vous le souhaitez), que vous receviez le code source ou pouviez l’obtenir si vous le voulez, que vous pouviez modifier le logiciel ou en utiliser toute partie dans de nouveaux logiciels libres, et que vous sachiez que vous avez le droit de faire tout ceci.
Pour protéger vos droits, nous avons besoin d’empêcher que d’autres vous restreignent ces droits ou vous demande de leur abandonner ces droits. En conséquence, vous avez certaines responsabilités si vous distribuez des copies d’un tel programme ou si vous le modifiez: les responsabilités de respecter la liberté des autres.
Par exemple, si vous distribuez des copies d’un tel programme, que ce soit gratuit ou contre un paiement, vous devez accorder aux Destinataires les mêmes libertés que vous avez reçues. Vous devez aussi vous assurer qu’eux aussi reçoivent ou peuvent recevoir son code source. Et vous devez leur montrer les termes de cette licence afin qu’ils connaissent leurs droits.
Les développeurs qui utilisent la GPL GNU protègent vos droits en deux étapes : (1) ils affirment leur droits d’auteur (**“**copyright**”**) sur le logiciel, et (2) vous accordent cette Licence qui vous donne la permission légale de le copier, le distribuer et/ou le modifier.
Pour la protection des développeurs et auteurs, la GPL stipule clairement qu’il n’y a pas de garantie pour ce logiciel libre. Aux fins à la fois des utilisateurs et auteurs, la GPL requière que les versions modifiées soient marquées comme changées, afin que leurs problèmes ne soient pas attribués de façon erronée aux auteurs des versions précédentes.
Certains dispositifs sont conçus pour empêcher l’accès des utilisateurs à l’installation ou l’exécution de versions modifiées du logiciel à l’intérieur de ces dispositifs, alors que les fabricants le peuvent. Ceci est fondamentalement incompatible avec le but de protéger la liberté des utilisateurs de modifier le logiciel. L’aspect systématique de tels abus se produit dans le secteur des produits destinés aux utilisateurs individuels, ce qui est précisément ce qui est le plus inacceptable. Aussi, nous avons conçu cette version de la GPL pour prohiber cette pratique pour ces produits. Si de tels problèmes surviennent dans d’autres domaines, nous nous tenons prêt à étendre cette restriction à ces domaines dans de futures versions de la GPL, autant qu’il sera nécessaire pour protéger la liberté des utilisateurs.
Finalement, chaque programme est constamment menacé par les brevets logiciels. Les États ne devraient pas autoriser de tels brevets à restreindre le développement et l’utilisation de logiciels libres sur des ordinateurs d’usage général; mais dans ceux qui le font, nous voulons spécialement éviter le danger que les brevets appliqués à un programme libre puisse le rendre effectivement propriétaire. Pour empêcher ceci, la GPL assure que les brevets ne peuvent être utilisés pour rendre le programme non-libre.
Les termes précis et conditions concernant la copie, la distribution et la modification suivent.

TERMES ET CONDITIONS
Article 0. Définitions.
"Cette Licence" se réfère à la version 3 de la **“**GNU General Public License**”** (le texte original en anglais).
"Droit d’Auteur" signifie aussi les droits du “copyright” ou voisins qui s’appliquent à d’autres types d’œuvres, tels que celles sur les masques de semi-conducteurs.
"Le Programme" se réfère à toute œuvre qui peut être sujette au Droit d’Auteur (“copyright”) et dont les droits d’utilisation sont concédés en vertu de cette Licence. Chacun des Licenciés, à qui cette Licence est concédée, est désigné par "vous." Les "Licenciés" et les "Destinataires" peuvent être des personnes physiques ou morales (individus ou organisations).
"Modifier" une œuvre signifie en obtenir une copie et adapter tout ou partie de l’œuvre d’une façon qui nécessite une autorisation d’un titulaire de Droit d’Auteur, autre que celle permettant d’en produire une copie conforme. L’œuvre résultante est appelée une "version modifiée" de la précédente œuvre, ou une œuvre "basée sur" la précédente œuvre.
Une "Œuvre Couverte" signifie soit le Programme non modifié soit une œuvre basée sur le Programme.
"Propager" une œuvre signifie faire quoi que ce soit avec elle qui, sans permission, vous rendrait directement ou indirectement responsable d’un délit de contrefaçon suivant les lois relatives au Droit d’Auteur, à l’exception de son exécution sur un ordinateur ou de la modification d’une copie privée. La propagation inclue la copie, la distribution (avec ou sans modification), la mise à disposition envers le public, et aussi d'autres activités dans certains pays.
"Acheminer" une œuvre signifie tout moyen de propagation de celle-ci qui permet à d’autres parties d’en réaliser ou recevoir des copies. La simple interaction d’un utilisateur à travers un réseau informatique, sans transfert effectif d’une copie, ne constitue pas un acheminement.
Une interface utilisateur interactive affiche des "Notices Légales Appropriées" quand elle comprend un dispositif convenable, bien visible et évident qui (1) affiche une notice appropriée sur les droits d’auteur et (2) informe l’utilisateur qu’il n’y a pas de garantie pour l’œuvre (sauf si des garanties ont été fournies hors du cadre de cette Licence), que les licenciés peuvent acheminer l’œuvre sous cette Licence, et comment consulter une copie de cette Licence. Si l’interface présente une liste de commandes utilisateur ou d’options, tel qu’un menu, un élément évident dans la liste présentée remplit ce critère.
Article 1. Code source.
Le "code source" d’une œuvre signifie la forme préférée de l’œuvre qui permet ou facilite les modifications de celle-ci. Le "code objet" d’une œuvre signifie toute forme de l’œuvre qui n’en est pas le code source.
Une "Interface Standard" signifie une interface qui est soit celle d’une norme officielle définie par un organisme de normalisation reconnu ou, dans le cas des interfaces spécifiées pour un langage de programmation particulier, une interface largement utilisée parmi les développeurs qui travaillent dans ce langage.
Les "Bibliothèques Système" d’une œuvre exécutable incluent tout ce qui, en dehors de l’œuvre dans son ensemble, (a) est inclus dans la forme usuelle de paquetage d’un Composant Majeur mais ne fait pas partie de ce Composant Majeur et (b) sert seulement à permettre l’utilisation de l’œuvre avec ce Composant Majeur ou à mettre en œuvre une Interface Standard pour laquelle une mise en œuvre est disponible au public sous forme de code source; un "Composant Majeur" signifie, dans ce contexte, un composant majeur essentiel (noyau, système de fenêtrage, etc.) du système d’exploitation (le cas échéant) d’un système sur lequel l’œuvre exécutable fonctionne, ou bien un compilateur utilisé pour produire le code objet de l’œuvre, ou un interprète de code objet utilisé pour exécuter celui-ci.
Le "Source Correspondant" d’une œuvre sous forme de code objet signifie l’ensemble des codes sources nécessaires pour générer, installer et (dans le cas d’une œuvre exécutable) exécuter le code objet et modifier l’œuvre, y compris les scripts pour contrôler ces activités. Cependant, cela n’inclue pas les Bibliothèques Système de l’œuvre, ni les outils d’usage général ou les programmes libres généralement disponibles qui peuvent être utilisés sans modification pour achever ces activités mais ne sont pas partie de cette œuvre. Par exemple le Source Correspondant inclut les fichiers de définition d’interfaces associés aux fichiers sources de l’œuvre, et le code source des bibliothèques partagées et des sous-routines liées dynamiquement, pour lesquelles l’œuvre est spécifiquement conçue pour les requérir via, par exemple, des communications de données ou contrôles de flux internes entre ces sous-programmes et d’autres parties de l’œuvre.
Le Source Correspondant n’a pas besoin d’inclure tout ce que les utilisateurs peuvent régénérer automatiquement à partir d’autres parties du Source Correspondant.
Le Source Correspondant pour une œuvre sous forme de code source est cette même œuvre.
Article 2. Permissions de base.
Tous les droits accordés suivant cette Licence le sont jusqu’au terme des Droits d’Auteur (“copyright”) sur le Programme, et sont irrévocables pourvu que les conditions établies soient remplies. Cette Licence affirme explicitement votre permission illimitée d’exécuter le Programme non modifié. La sortie produite par l’exécution d’une Œuvre Couverte n’est couverte par cette Licence que si cette sortie, étant donné leur contenu, constitue une Œuvre Couverte. Cette Licence reconnaît vos propres droits d’usage raisonnable (“fair use” en législation des États-Unis d’Amérique) ou autres équivalents, tels qu’ils sont pourvus par la loi applicable sur le Droit d’Auteur (“copyright”).
Vous pouvez créer, exécuter et propager sans condition des Œuvres Couvertes que vous n’acheminez pas, aussi longtemps que votre licence demeure en vigueur. Vous pouvez acheminer des Œuvres Couvertes à d’autres personnes dans le seul but de leur faire réaliser des modifications à votre usage exclusif, ou pour qu’ils vous fournissent des facilités vous permettant d’exécuter ces œuvres, pourvu que vous vous conformiez aux termes de cette Licence lors de l’acheminement de tout matériel dont vous ne contrôlez pas le Droit d’Auteur (“copyright”). Ceux qui, dès lors, réalisent ou exécutent pour vous les Œuvres Couvertes ne doivent alors le faire qu’exclusivement pour votre propre compte, sous votre direction et votre contrôle, suivant des termes qui leur interdisent de réaliser, en dehors de leurs relations avec vous, toute copie de votre matériel soumis au Droit d’Auteur.
L’acheminement dans toutes les autres circonstances n’est permis que selon les conditions établies ci-dessous. La concession de sous-licences n’est pas autorisé; l’article 10 rend cet usage non nécessaire.
Article 3. Protection des droits légaux des utilisateurs envers les lois anti-contournement.
Aucune Œuvre Couverte ne doit être vue comme faisant partie d’une mesure technologique effective selon toute loi applicable remplissant les obligations prévues à l’article 11 du traité international sur le droit d’auteur adopté à l’OMPI le 20 décembre 1996, ou toutes lois similaires qui prohibent ou restreignent le contournement de telles mesures.
Si vous acheminez une Œuvre Couverte, vous renoncez à tout pouvoir légal d’interdire le contournement des mesures technologiques dans tous les cas où un tel contournement serait effectué en exerçant les droits prévus dans cette Licence pour cette Œuvre Couverte, et vous déclarez rejeter toute intention de limiter l’opération ou la modification de l’Œuvre, en tant que moyens pour renforcer, à l’encontre des utilisateurs de cette Œuvre, vos droits légaux ou ceux de tierces parties d’interdire le contournement desdites mesures technologiques.
Article 4. Acheminement des copies conformes.
Vous pouvez acheminer des copies conformes du code source du Programme tel que vous l’avez reçu, sur n’importe quel support, pourvu que vous publiiez scrupuleusement et de façon appropriée sur chaque copie une notice de Droit d’Auteur appropriée; gardez intactes toutes les notices établissant que cette Licence et tous les termes additionnels non permissifs ajoutés en accord avec l’article 7 s’appliquent à ce code; et donnez à chacun des Destinataires une copie de cette Licence en même temps que le Programme.
Vous pouvez facturer un prix quelconque, y compris gratuit, pour chacune des copies que vous acheminez, et vous pouvez offrir une protection additionnelle de support ou de garantie en échange d’un paiement.
Article 5. Acheminement des versions sources modifiées.
Vous pouvez acheminer une œuvre basée sur le Programme, ou bien les modifications pour le produire à partir du Programme, sous la forme de code source suivant les termes de l’article 4, pourvu que vous satisfassiez aussi à chacune des conditions requises suivantes :
- a) L’œuvre doit comporter des notices évidentes établissant que vous l’avez modifiée et donnant la date correspondante.
- b) L’œuvre doit comporter des notices évidentes établissant qu’elle est éditée selon cette Licence et les conditions ajoutées d’après l’article 7. Cette obligation vient modifier l’obligation de l’article 4 de "garder intactes toutes les notices."
- c) Vous devez licencier l’œuvre entière, comme un tout, suivant cette Licence à quiconque entre en possession d’une copie. Cette Licence s’appliquera en conséquence, avec les termes additionnels applicables prévus par l’article 7, à la totalité de l’œuvre et chacune de ses parties, indépendamment de la façon dont elles sont empaquetées. Cette licence ne donne aucune permission de licencier l’œuvre d’une autre façon, mais elle n’invalide pas une telle permission que vous auriez reçue séparément.
- d) Si l’œuvre a des interfaces utilisateurs interactives, chacune doit afficher les Notices Légales Appropriées; cependant si le Programme a des interfaces qui n’affichent pas les Notices Légales Appropriées, votre œuvre n’a pas à les modifier pour qu’elles les affichent.

Une compilation d’une Œuvre Couverte avec d’autres œuvres séparées et indépendantes, qui ne sont pas par leur nature des extensions de l’Œuvre Couverte, et qui ne sont pas combinés avec elle de façon à former un programme plus large, dans ou sur un volume de stockage ou un support de distribution, est appelé un "agrégat" si la compilation et son Droit d’Auteur résultant ne sont pas utilisés pour limiter l’accès ou les droits légaux des utilisateurs de la compilation en deçà de ce que permettent les œuvres individuelles. L’inclusion d’une Œuvre Couverte dans un agrégat ne cause pas l’application de cette Licence aux autres parties de l’agrégat.
Article 6. Acheminement des formes non sources.
Vous pouvez acheminer sous forme de code objet une Œuvre Couverte suivant les termes des articles 4 et 5, pourvu que vous acheminiez également suivant les termes de cette Licence le Source Correspondant lisible par une machine, d’une des façons suivantes :
- a) Acheminer le code objet sur, ou inclus dans, un produit physique (y compris un support de distribution physique), accompagné par le Source Correspondant fixé sur un support physique durable habituellement utilisé pour les échanges de logiciels.
- b) Acheminer le code objet sur, ou inclus dans, un produit physique (y compris un support de distribution physique), accompagné d’une offre écrite, valide pour au moins trois années et valide pour aussi longtemps que vous fournissiez des pièces de rechange ou un support client pour ce modèle de produit, afin de donner à quiconque possède le code objet soit (1) une copie du Source Correspondant à tout logiciel dans ce produit qui est couvert par cette Licence, sur un support physique durable habituellement utilisé pour les échanges de logiciels, pour un prix non supérieur au coût raisonnable de la réalisation physique de l’acheminement de la source, ou soit (2) un accès permettant de copier le Source Correspondant depuis un serveur réseau sans frais.
- c) Acheminer des copies individuelles du code objet avec une copie de l’offre écrite de fournir le Source Correspondant. Cette alternative est permise seulement occasionnellement et non-commercialement, et seulement si vous avez reçu le code objet avec une telle offre, en accord avec l’article 6 alinéa b.
- d) Acheminer le code objet en offrant un accès depuis un emplacement désigné (gratuit ou contre facturation) et offrir un accès équivalent au Source Correspondant de la même façon via le même emplacement et sans facturation supplémentaire. Vous n’avez pas besoin d’obliger les Destinataires à copier le Source Correspondant en même temps que le code objet. Si l’emplacement pour copier le code objet est un serveur réseau, le Source Correspondant peut être sur un serveur différent (opéré par vous ou par un tiers) qui supporte des facilités équivalentes de copie, pourvu que vous mainteniez des directions claires à proximité du code objet indiquant où trouver le Source Correspondant. Indépendamment de quel serveur héberge le Source Correspondant, vous restez obligé de vous assu-

rer qu'il reste disponible aussi longtemps que nécessaire pour satisfaire à ces obligations.

■ e) Acheminer le code objet en utilisant une transmission d'égal-à-égal, pourvu que vous informiez les autres participants sur l'endroit où le code objet et le Source Correspondant de l'œuvre sont offerts sans frais au public général suivant l'article 6 alinéa d.

Une portion séparable du code objet, dont le code source est exclu du Source Correspondant en tant que Bibliothèque Système, n'a pas besoin d'être incluse dans l'acheminement de l'œuvre sous forme de code objet.

Un "Produit Utilisateur" est soit (1) un "Produit de Consommation," ce qui signifie toute propriété personnelle tangible normalement utilisée à des fins personnelles, familiales ou relatives au foyer, soit (2) toute chose conçue ou vendue pour l'incorporation dans un lieu d'habitation. Pour déterminer si un produit constitue un Produit de Consommation, les cas ambigus sont résolus en fonction de la couverture. Pour un produit particulier reçu par un utilisateur particulier, l'expression "normalement utilisée" ci-avant se réfère à une utilisation typique ou l'usage commun de produits de même catégorie, indépendamment du statut de cet utilisateur particulier ou de la façon spécifique dont cet utilisateur particulier utilise effectivement ou s'attend lui-même ou est attendu à utiliser ce produit. Un produit est un Produit de Consommation indépendamment du fait que ce produit a ou n'a pas d'utilisations substantielles commerciales, industrielles ou hors Consommation, à moins que de telles utilisations représentent le seul mode significatif d'utilisation du produit.

Les "Informations d'Installation" d'un Produit Utilisateur signifient toutes les méthodes, procédures, clés d'autorisation ou autres informations requises pour installer et exécuter des versions modifiées d'une Œuvre Couverte dans ce Produit Utilisateur à partir d'une version modifiée de son Source Correspondant. Les informations qui suffisent à assurer la continuité de fonctionnement du code objet modifié ne doivent en aucun cas être empêchées ou interférées du seul fait qu'une modification a été effectuée.

Si vous acheminez le code objet d'une Œuvre Couverte dans, ou avec, ou spécifiquement pour l'utilisation dans, un Produit Utilisateur et si l'acheminement se produit en tant qu'élément d'une transaction dans laquelle le droit de possession et d'utilisation du Produit Utilisateur est transféré au Destinataire définitivement ou pour un terme fixé (indépendamment de la façon dont la transaction est caractérisée), le Source Correspondant acheminé selon cet article-ci doit être accompagné des Informations d'Installation. Mais cette obligation ne s'applique pas si ni vous ni aucune tierce partie ne détient la possibilité d'installer un code objet modifié sur le Produit Utilisateur (par exemple, l'œuvre a été installée en mémoire morte).

L'obligation de fournir les Informations d'Installation n'inclue pas celle de continuer à fournir un service de support, une garantie ou des mises à jour pour une œuvre qui a été modifiée ou installée par le Destinataire, ou pour le Produit Utilisateur dans lequel elle a été modifiée ou installée. L'accès à un réseau peut être rejeté quand la modification elle-même affecte matériellement et défavorablement les opérations du réseau ou viole les règles et protocoles de communication au travers du réseau.

Le Source Correspondant acheminé et les Informations d'Installation fournies, en accord avec cet article, doivent être dans un format publiquement documenté (et dont une implémentation est disponible auprès du public sous forme de code source) et ne doit nécessiter aucune clé ou mot de passe spécial pour le dépaquetage, la lecture ou la copie.

Article 7. Termes additionnels.

Les « permissions additionnelles » désignent les termes qui supplémentent ceux de cette Licence en émettant des exceptions à l'une ou plusieurs de ses conditions. Les permissions additionnelles qui sont applicables au Programme entier doivent être traitées comme si elles étaient incluent dans cette Licence, dans les limites de leur validité suivant la loi applicable. Si des permissions additionnelles s'appliquent seulement à une partie du Programme, cette partie peut être utilisée séparément suivant ces permissions, mais le Programme tout entier reste gouverné par cette Licence sans regard aux permissions additionnelles.

Quand vous acheminez une copie d'une Œuvre Couverte, vous pouvez à votre convenance ôter toute permission additionnelle de cette copie, ou de n'importe quelle partie de celui-ci. (Des permissions additionnelles peuvent être rédigées de façon à requérir leur propre suppression dans certains cas où vous modifiez l'œuvre.) Vous pouvez placer les permissions additionnelles sur le matériel acheminé, ajoutées par vous à une Œuvre Couverte pour laquelle vous avez ou pouvez donner les permissions de Droit d'Auteur ("copyright") appropriées.

Nonobstant toute autre clause de cette Licence, pour tout constituant que vous ajoutez à une Œuvre Couverte, vous pouvez (si autorisé par les titulaires de Droit d'Auteur pour ce constituant) supplémenter les termes de cette Licence avec des termes :

■ a) qui rejettent la garantie ou limitent la responsabilité de façon différente des termes des articles 15 et 16 de cette Licence; ou

■ b) qui requièrent la préservation de notices légales raisonnables spécifiées ou les attributions d'auteur dans ce constituant ou dans les Notices Légales Appropriées affichées par les œuvres qui le contiennent; ou

■ c) qui prohibent la représentation incorrecte de l'origine de ce constituant, ou qui requièrent que les versions modifiées d'un tel constituant soient marquées par des moyens raisonnables comme différentes de la version originale; ou

■ d) qui limitent l'usage à but publicitaire des noms des concédants de licence et des auteurs du constituant; ou

■ e) qui refusent à accorder des droits selon la législation relative aux marques commerciales, pour l'utilisation dans des noms commerciaux, marques commerciales ou marques de services; ou

■ f) qui requièrent l'indemnisation des concédants de licences et auteurs du constituant par quiconque achemine ce constituant (ou des versions modifiées de celui-ci) en assumant contractuellement la responsabilité envers le Destinataire, pour toute responsabilité que ces engagements contractuels imposent directement à ces octroyants de licences et auteurs.

Tous les autres termes additionnels non permissifs sont considérés comme des « restrictions avancées » dans le sens de l'article 10. Si le Programme tel que vous l'avez reçu, ou toute partie de celui-ci, contient une notice établissant qu'il est gouverné par cette Licence en même temps qu'un terme qui est une restriction avancée, vous pouvez ôter ce terme. Si un document de licence contient une restriction avancée mais permet la reconcession de licence ou l'acheminement suivant cette Licence, vous pouvez ajouter une Œuvre Couverte constituante gouvernée par les termes de ce document de licence, pourvu que la restriction avancée ne survit pas à une telle cession de licence ou un tel acheminement.

Si vous ajoutez des termes à une Œuvre Couverte en accord avec cet article, vous devez placer, dans les fichiers sources appropriés, une déclaration des termes additionnels qui s'appliquent à ces fichiers, ou une notice indiquant où trouver les termes applicables.

Les termes additionnels, qu'ils soient permissifs ou non permissifs, peuvent être établis sous la forme d'une licence écrite séparément, ou établis comme des exceptions; les obligations ci-dessus s'appliquent dans chacun de ces cas.

Article 8. Terminaison.

Vous ne pouvez ni propager ni modifier une Œuvre Couverte autrement que suivant les termes de cette Licence. Toute autre tentative de le propager ou le modifier est nulle et terminera automatiquement vos droits selon cette Licence (y compris toute licence de brevet accordée selon le troisième paragraphe de l'article 11).

Cependant, si vous cessez toute violation de cette Licence, alors votre licence depuis un titulaire de Droit d'Auteur ("copyright") est réinstaurée (a) à titre provisoire à moins que et jusqu'à ce que le titulaire de Droit d'Auteur termine finalement et explicitement votre licence, et (b) de façon permanente si le titulaire de Droit d'Auteur ne parvient pas à vous notifier de la violation par quelque moyen raisonnable dans les soixante (60) jours après la cessation.

De plus, votre licence depuis un titulaire particulier de Droit d'Auteur est réinstaurée de façon permanente si ce titulaire vous a notifié de la violation par quelque moyen raisonnable, et si c'est la première fois que vous avez reçu une notification de violation de cette Licence (pour une œuvre quelconque) depuis ce titulaire de Droit d'Auteur, et si vous résolvez la violation dans les trente (30) jours qui suivent votre réception de la notification.

La terminaison de vos droits suivant cette section ne terminera pas les licences des parties qui ont reçu des copies ou droits de votre part suivant cette Licence. Si vos droits ont été terminés et non réinstaurés de façon permanente, vous n'êtes plus qualifié à recevoir de nouvelles licences pour les mêmes constituants selon l'article 10.

Article 9. Acceptation non requise pour obtenir des copies.

Vous n'êtes pas obligé d'accepter cette licence afin de recevoir ou exécuter une copie du Programme. La propagation asservie d'une Œuvre Couverte qui se produit simplement en conséquence d'une transmission d'égal-à-égal pour recevoir une copie ne nécessite pas l'acceptation. Cependant, rien d'autre que cette Licence ne vous accorde la permission de propager ou modifier une quelconque Œuvre Couverte. Ces actions enfreignent le Droit d'Auteur si vous n'acceptez pas cette Licence. Par conséquent, en modifiant ou propageant une Œuvre Couverte, vous indiquez votre acceptation de cette Licence pour agir ainsi.

Article 10. Cession automatique de Licence aux Destinataires et intermédiaires.

Chaque fois que vous acheminez une Œuvre Couverte, le Destinataire reçoit automatiquement une licence de la part des concédants originaux, pour exécuter, modifier et propager cette œuvre, suivant les termes de cette Licence. Vous n'êtes pas responsable du renforcement de la conformation des tierces parties aux termes de cette Licence.

Une "transaction d'entité" désigne une transaction qui transfère le contrôle d'une organisation, ou de substantiellement tous ses actifs, ou la subdivision d'une organisation, ou la fusion de plusieurs organisations. Si la propagation d'une Œuvre Couverte résulte d'une transaction d'entité, chaque partie à cette transaction qui reçoit une copie de l'œuvre reçoit aussi les licences pour l'œuvre que le prédécesseur intéressé à cette partie avait ou pourrait donner selon le paragraphe précédent, plus un droit de possession du Source Correspondant de cette œuvre depuis le prédécesseur intéressé si ce prédécesseur en dispose ou peut l'obtenir par des efforts raisonnables.

Vous ne pouvez imposer aucune restriction avancée dans l'exercice des droits accordés ou affirmés selon cette Licence. Par exemple, vous ne pouvez imposer aucun paiement pour la licence, aucune royaltie, ni aucune autre charge pour l'exercice des droits accordés selon cette Licence; et vous ne pouvez amorcer aucun litige judiciaire (y compris une réclamation croisée ou contre-réclamation dans un procès) sur l'allégation qu'une revendication de brevet est enfreinte par la réalisation, l'utilisation, la vente, l'offre de vente, ou l'importation du Programme ou d'une quelconque portion de celui-ci.

Article 11. Brevets.

Un "contributeur" est un titulaire de Droit d'Auteur ("**copyright**") qui autorise l'utilisation selon cette Licence du Programme ou du travail sur lequel le Programme est basé. Le travail ainsi soumis à licence est appelé la "version contributive" de ce contributeur.

Les « revendications de brevet essentielles » sont toutes les revendications de brevets détenues ou contrôlées par le contributeur, qu'elles soient déjà acquises par lui ou acquises subséquemment, qui pourraient être enfreintes de quelque manière, permises par cette Licence, sur la réalisation, l'utilisation ou la vente de la version contributive de celui-ci. Aux fins de cette définition, le "contrôle" inclue le droit de concéder des sous-licences de brevets d'une manière consistante, nécessaire et suffisante, avec les obligations de cette Licence.

Chaque contributeur vous accorde une licence de brevet non exclusive, mondiale et libre de toute royaltie, selon les revendications de brevet essentielles, pour réaliser, utiliser, vendre, offrir à la vente, importer et autrement exécuter, modifier et propager les contenus de sa version contributive.

Dans les trois paragraphes suivants, une « licence de brevet » désigne tous les accords ou engagements exprimés, quel que soit le nom que vous lui donnez, de ne pas mettre en vigueur un brevet (telle qu'une permission explicite pour mettre en pratique un brevet, ou un accord pour ne pas poursuivre un Destinataire pour cause de violation de brevet). "Accorder" une telle licence de brevet à une partie signifie conclure un tel accord ou engagement à ne pas faire appliquer le brevet à cette partie.

Si vous acheminez un Travail Couvert, dépendant en connaissance d'une licence de brevet, et si le Source Correspondant du travail n'est pas disponible à quiconque copie, sans frais et suivant les termes de cette Licence, à travers un serveur réseau publiquement accessible ou tout autre moyen immédiatement accessible, alors vous devez soit (1) rendre la Source Correspondante ainsi disponible, soit (2) vous engager à vous priver pour vous-même du bénéfice de la licence de brevet pour ce travail particulier, soit (3) vous engager, d'une façon consistante avec les obligations de cette Licence, à étendre la licence de brevet aux Destinataires de ce travail. "Dépendant en connaissance" signifie que vous avez effectivement connaissance que, selon la licence de brevet, votre acheminement du Travail Couvert dans un pays, ou l'utilisation du Travail Couvert par votre Destinataire dans un pays, enfreindrait un ou plusieurs brevets identifiables dans ce pays où vous avez des raisons de penser qu'ils sont valides.

Si, conformément à ou en liaison avec une même transaction ou un même arrangement, vous acheminez, ou propagez en procurant un acheminement de, un Travail Couvert et accordez une licence de brevet à l'une des parties recevant le Travail Couvert pour lui permettre d'utiliser, propager, modifier ou acheminer une copie spécifique du Travail Couvert, alors votre accord est automatiquement étendu à tous les Destinataires du Travail Couvert et des travaux basés sur celui-ci.

Une licence de brevet est "discriminatoire" si, dans le champ de sa couverture, elle n'inclut pas un ou plusieurs des droits qui sont spécifiquement accordés selon cette Licence, ou en prohibe l'exercice, ou est conditionnée par le non-exercice d'un ou plusieurs de ces droits. Vous ne pouvez pas acheminer un Travail Couvert si vous êtes partie à un arrangement selon lequel une partie tierce exerçant son activité dans la distribution de logiciels et à laquelle vous effectuez un paiement fondé sur l'étendue de votre activité d'acheminement du travail, et selon lequel la partie tierce accorde, à une quelconque partie qui recevrait depuis vous le Travail Couvert, une licence de brevet discriminatoire (a) en relation avec les copies du Travail Couvert acheminées par vous (ou les copies réalisées à partir de ces copies), ou (b) avant tout destinée et en relation avec des produits spécifiques ou compilations contenant le Travail Couvert, à moins que vous ayez conclu cet arrangement ou que la licence de brevet ait été accordée avant le 28 mars 2007.

Rien dans cette Licence ne devrait être interprété comme devant exclure ou limiter toute licence implicite ou d'autres moyens de défense à une infraction qui vous seraient autrement disponible selon la loi applicable relative aux brevets.

Article 12. Non abandon de la liberté des autres.

Si des conditions vous sont imposées (que ce soit par décision judiciaire, par un accord ou autrement) qui contredisent les conditions de cette Licence, elles ne vous excusent pas des conditions de cette Licence. Si vous ne pouvez pas acheminer une Œuvre Couverte de façon à satisfaire simultanément vos obligations suivant cette Licence et toutes autres obligations pertinentes, alors en conséquence vous ne pouvez pas du tout l'acheminer. Par exemple, si vous avez un accord sur des termes qui vous obligent à collecter pour le réacheminement des royalties depuis ceux à qui vous acheminez le Programme, la seule façon qui puisse vous permettre de satisfaire à la fois à ces termes et ceux de cette Licence sera de vous abstenir entièrement d'acheminer le Programme.

Article 13. Utilisation avec la Licence Générale Publique Affero GNU.

Nonobstant toute autre clause de cette Licence, vous avez la permission de lier ou combiner toute Œuvre Couverte avec une œuvre placée sous la version 3 de la Licence Générale Publique GNU Affero ("GNU Affero General Public License") en une seule œuvre combinée, et d'acheminer l'œuvre résultante. Les termes de cette Licence continueront à s'appliquer à la partie formant une Œuvre Couverte, mais les obligations spéciales de la Licence Générale Publique GNU Affero, article 13, concernant l'interaction à travers un réseau, s'appliqueront à la combinaison en tant que telle.

Article 14. Versions révisées de cette Licence.

La Free Software Foundation peut publier des versions révisées et/ou nouvelles de la Licence Publique Générale GNU ("GNU General Public License") de temps en temps. De telles version nouvelles resteront similaires dans l'esprit avec la présente version, mais peuvent différer dans le détail afin de traiter de nouveaux problèmes ou préoccupations.

Chaque version reçoit un numéro de version distinctif. Si le Programme indique qu'une version spécifique de la Licence Publique Générale GNU "ou toute version ultérieure" ("or any later version") s'applique à celui-ci, vous avez le choix de suivre soit les termes et conditions de cette version numérotée, soit ceux de n'importe quelle version publiée ultérieurement par la Free Software Foundation. Si le Programme n'indique pas une version spécifique de la Licence Publique Générale GNU, vous pouvez choisir l'une quelconque des versions qui ont été publiées par la Free Software Foundation.

Si le Programme spécifie qu'un intermédiaire peut décider quelles versions futures de la Licence Générale Publique GNU peut être utilisée, la déclaration publique d'acceptation d'une version par cet intermédiaire vous autorise à choisir cette version pour le Programme.

Des versions ultérieures de la licence peuvent vous donner des permissions additionnelles ou différentes. Cependant aucune obligation additionnelle n'est imposée à l'un des auteurs ou titulaires de Droit d'Auteur du fait de votre choix de suivre une version ultérieure.

Article 15. Déclaration d'absence de garantie.

Il n'y a aucune garantie pour le programme, dans les limites permises par la loi applicable. À moins que cela ne soit établi différemment par écrit, les propriétaires de droits et/ou les autres parties fournissent le programme "en l'état" sans garantie d'aucune sorte, qu'elle soit exprimée ou implicite, ceci comprenant, sans se limiter à celles-ci, les garanties implicites de commercialisabilité et d'adéquation à un objectif particulier. Vous assumez le risque entier concernant la qualité et les performances du programme. Dans l'éventualité où le programme s'avérerait défectueux, vous assumez les coûts de tous les services, réparations ou corrections nécessaires.

Article 16. Limitation de responsabilité.

En aucune autre circonstance que celles requises par la loi applicable ou accordées par écrit, un titulaire de droits sur le programme, ou tout autre partie qui modifie ou achemine le programme comme permis ci-dessus, ne peut être tenu pour responsable envers vous pour les dommages, incluant tout dommage général, spécial, accidentel ou induit survenant par suite de l'utilisation ou de l'incapacité d'utiliser le programme (y compris, sans se limiter à celles-ci, la perte de données ou l'inexactitude des données retournées ou les pertes subies par vous ou des parties tierces ou l'incapacité du programme à fonctionner avec tout autre programme), même si un tel titulaire ou toute autre partie a été avisé de la possibilité de tels dommages.

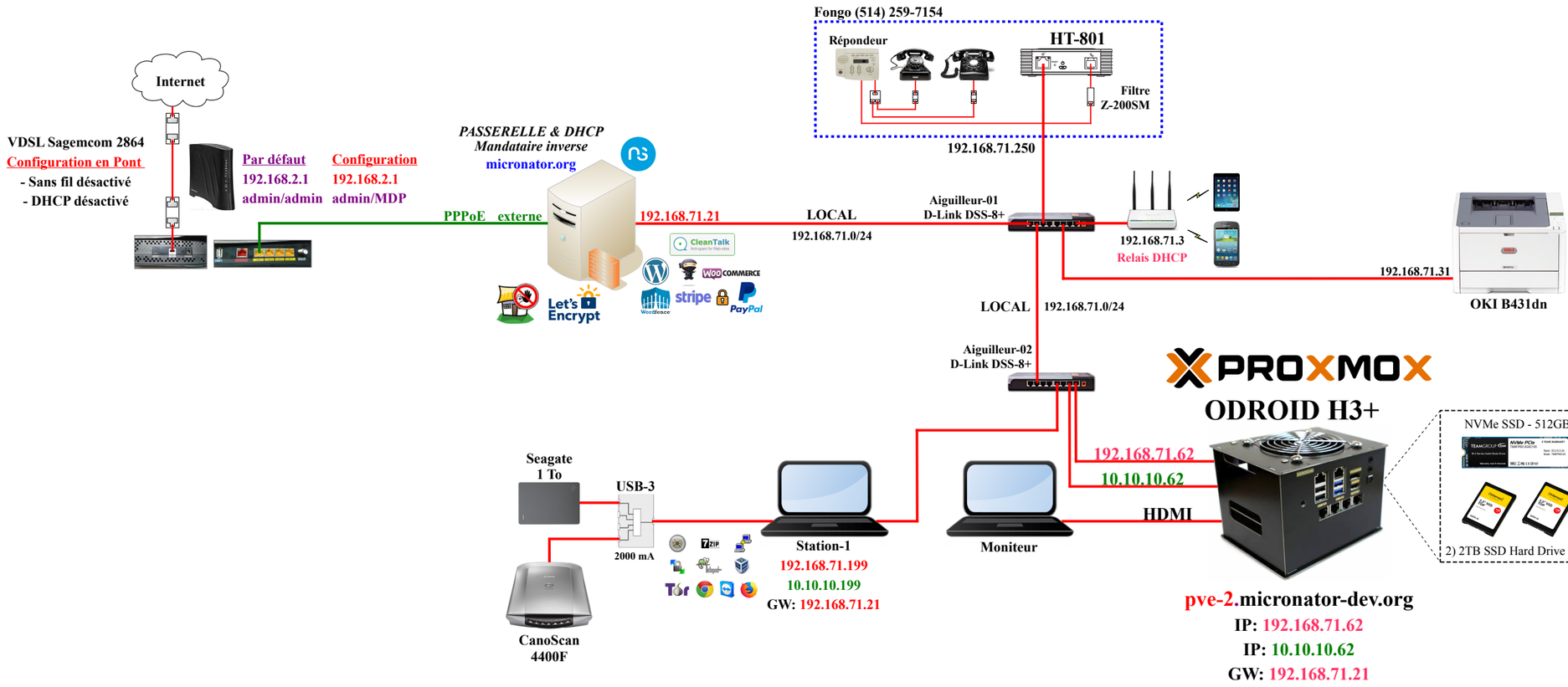
Article 17. Interprétation des sections 15 et 16.

Si la déclaration d'absence de garantie et la limitation de responsabilité fournies ci-dessus ne peuvent prendre effet localement selon leurs termes, les cours de justice qui les examinent doivent appliquer la législation locale qui approche au plus près possible une levée absolue de toute responsabilité civile liée au Programme, à moins qu'une garantie ou assumation de responsabilité accompagne une copie du Programme en échange d'un paiement.

FIN DES TERMES ET CONDITIONS.

Projet ISPconfig

Cours ISPconfig-101 *Cahier-01* Réseau initial 2024-11-28



Virtual Machines

Debian-12	Proxmox
NethServer-8	Flectra
Jitsi Meet	Forum
BackupPC	NodeBB
Dolibarr	Discourse
Odoo	Zabbix
Mediawiki	Zammad
Moodle	Mattermost
	Matrix

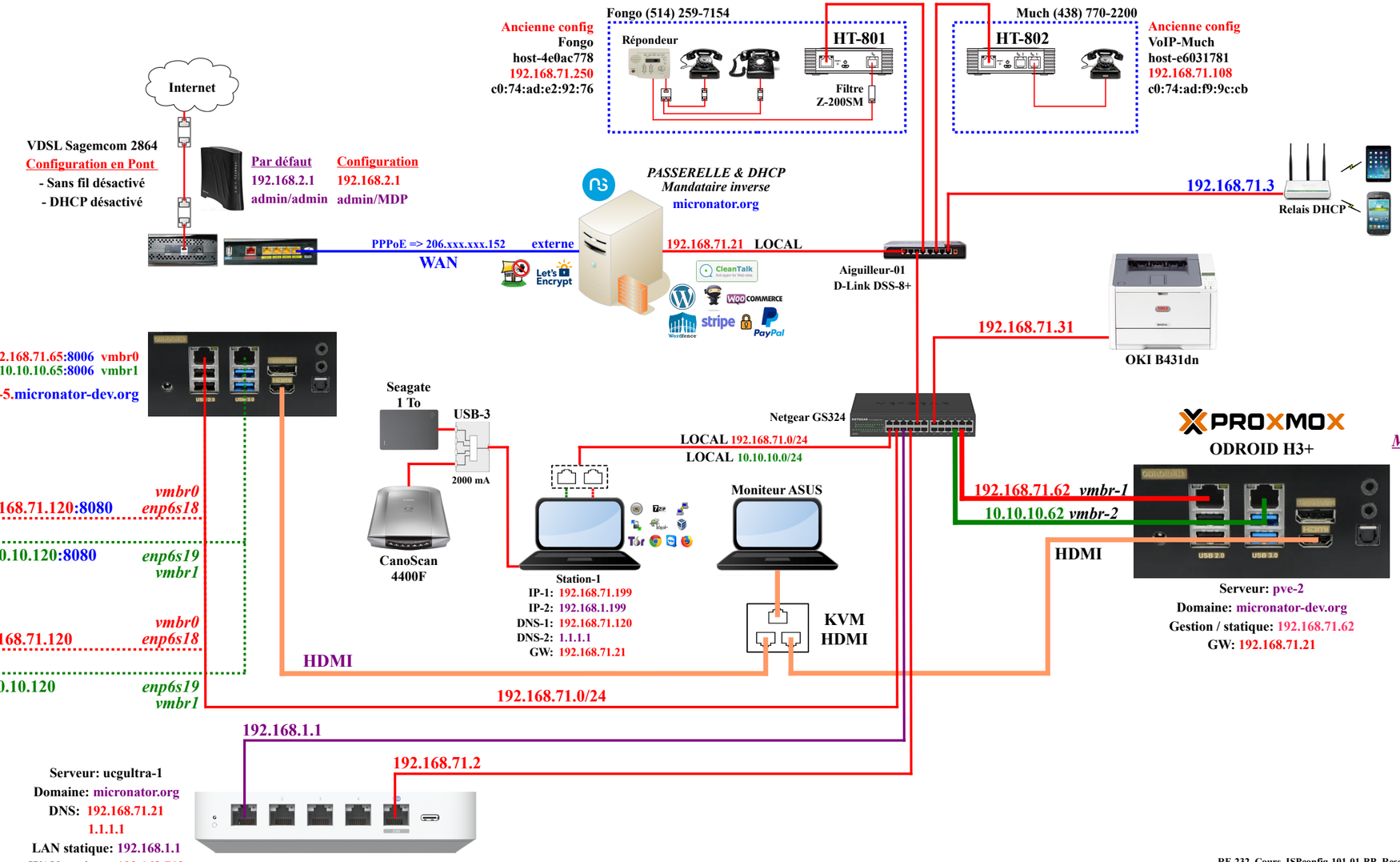
RF-232_Cours_ISPconfig-101-01-A_Reseau_Initial_2024-11-28_17h53.odg

Micronator, Projet ISPconfig				
Diagramme réseau				
Réseau initial				
Michel-André Robillard	Date de changement	Par	2024-11-26	**Projet ISPconfig**
Révision: 0.0.1	2024-11-26	MAR	Code	

Projet ISPconfig

Cours ISPconfig-101 *Cahier-01* PVE-5 et UCG-Ultra 2025-01-02

PPPoE NethServer



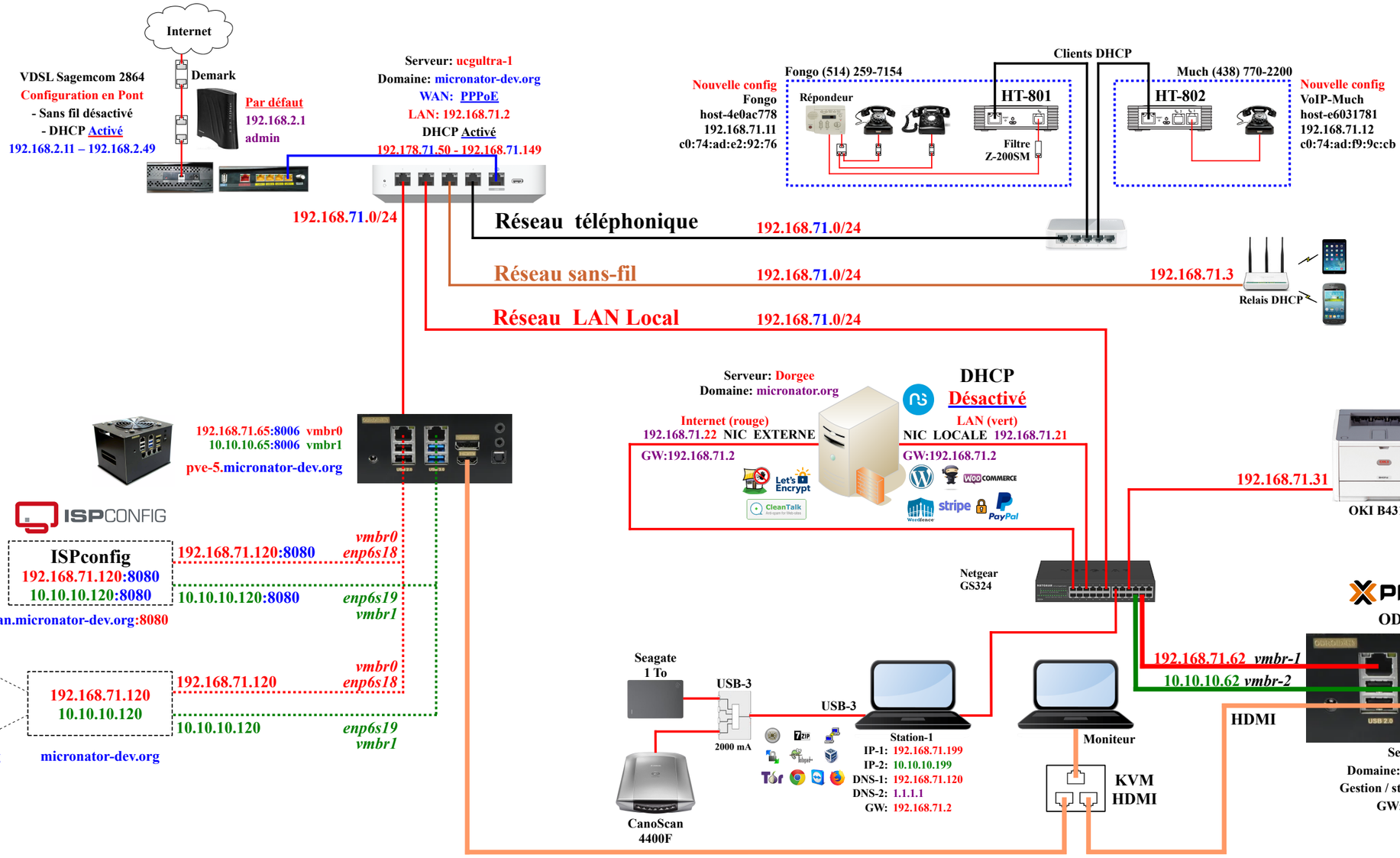
RF-232_Cours_ISPconfig-101-01-BB_Reseau_TEST_2025-01-02_10h38.odg

Micronator, Projet ISPconfig				
Diagramme réseau				
Proxmox PVE-5 & UCG Ultra				
Michel-André Robillard	Date de changement 2025-01-02	Par MAR	2024-08-23	Projet ISPconfig
Révision: 0.0.9			Code	

Projet ISPconfig

Cours ISPconfig-101 *Cahier-01* UCG-Ultra avec NethServer 2025-01-02

PPPoE UCG Ultra

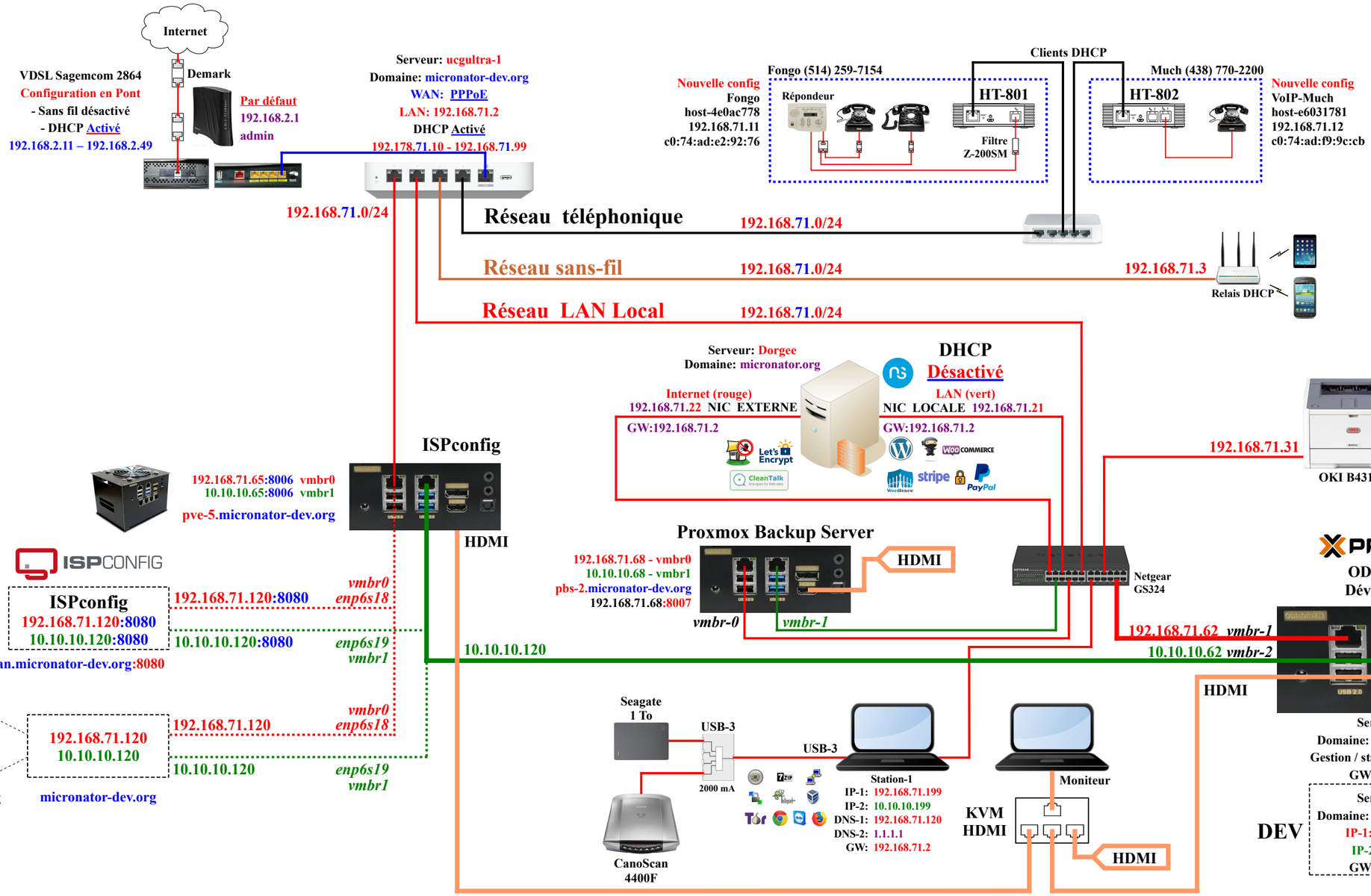


RF-232_Cours_ISPconfig-101-01-CC_Reseau_UCG_avec_NS-7.9_2025-01-02_10h55.odg

Micronator, Projet ISPconfig				
Diagramme réseau				
UCG Ultra avec NethServer-7.9				
Michel-André Robillard	Date de changement 2025-01-02	Par MAR	2024-08-26	Projet ISPconfig
Révision: 0.0.7			Code	

Projet ISPconfig

Cours Proxmox Backup Server-101 *Cahier-01-01* PBS virtuel 2025-01-02

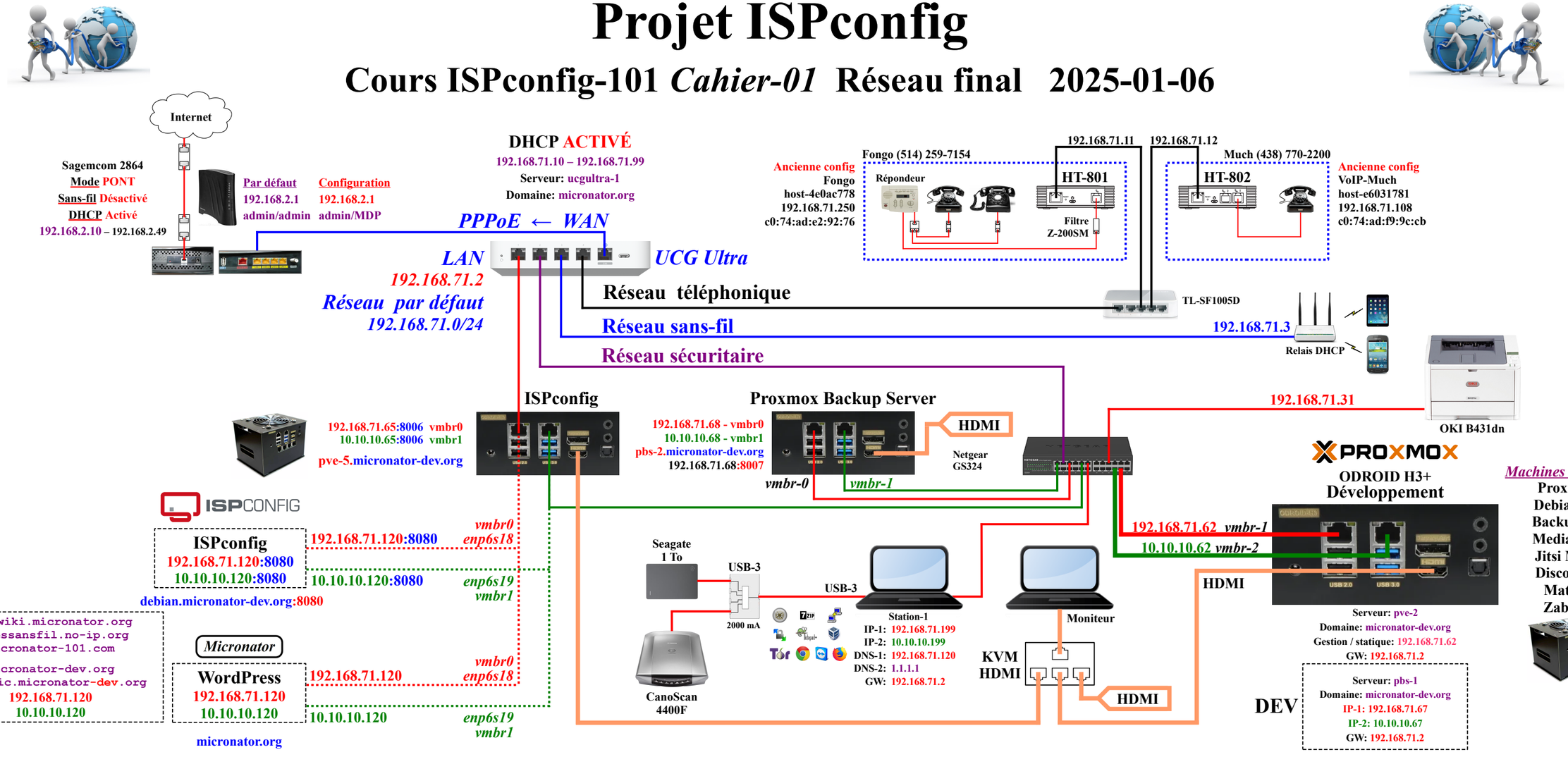


RF-232_Cours_Proxmox-Backup-Server-101-01-01_PBS-virtuel_2025-01-02_09h48.odg

Micronator, Projet ISPconfig				
Diagramme réseau				
Proxmox Backup Server				
Michel-André Robillard	Date de changement 2025-01-02	Par MAR	2024-12-16	Projet ISPconfig
Révision: 0.0.4			Code	

Projet ISPconfig

Cours ISPconfig-101 *Cahier-01* Réseau final 2025-01-06



UCG ULTRA - redirection de ports

443	192.168.71.120	https - debian.micronator=dev.org
~~980~~	~~192.168.71.21~~	~~gestion micronator.org~~
~~9090~~	~~192.168.71.21~~	~~gestion micronator.org~~
8006	192.168.71.65	gestion - pve-5.micronator-dev.org
8007	192.168.71.68	gestion - pbs-2.micronator-dev.org
8080	192.168.71.120	gestion - ispconfig.micronator-dev.org
110	192.168.71.120	POP3 IN sans chiffrement
995	192.168.71.120	POP3S IN SSL/TLS
143	192.168.71.120	IMAP IN STARTTLS ou SSL/TLS
993	192.168.71.120	IMAPS IN SSL/TLS
25	192.168.71.120	SMTP serveur à serveur RX/TX
465	192.168.71.120	SMTPS sécurisé Ancien
587	192.168.71.120	SMTPS sécurisé
2525	192.168.71.120	SMTP backup

Zones DNS du serveur de noms ISPconfig

		Adresse IP
Sites Web	micronator.org	192.168.71.120
	dokuwiki.micronator.org	192.168.71.120
	ainessansfil.no-ip.org	192.168.71.120
	micronator-101.com	192.168.71.120
UCG Ultra	ucgultra-1.micronator.org	192.168.71.2
Proxmox VE	pve-2.micronator-dev.org	192.168.71.62
Proxmox VE	pve-5.micronator-dev.org	192.168.71.65
Proxmox Backup Server	pbs-1.micronator-dev.org	192.168.71.67
Proxmox Backup Server	pbs-2.micronator-dev.org	192.168.71.68
Sites Web	micronator.org	192.168.71.120
	frederic.micronator-dev.org	192.168.71.120
Hôte d'IPSconfig	debian.micronator-dev-dev.org	192.168.71.120
IPSconfig	ispconfig.micronator-dev.org	192.168.71.120

Appareils – IP (Actuel / *Défaut*)

Sagemcom 2864	192.168.2.1 / *192.168.2.1*
ucgultra-1	192.168.72.2 / *192.168.1.1*
HT-801	192.168.71.11
HT-802	192.168.71.12
Imprimante OKI B431dn	192.168.71.31
Proxmox PVE-2	192.168.71.62
Proxmox PVE-5	192.168.71.65
Proxmox PBS-1 Proxmox PBS-2	VMBR-0 => 192.168.71.67/68 VMBR-1 => 10.10.10.67/68 GESTION => 192.168.71.67/68:8007
Poste de travail	192.168.1.1 192.168.71.199

RF-232_Cours_ISPconfig-101-01-DD_Reseau_UCG_FINAL_2025-01-06_15h12.odg

Micronator, Projet ISPconfig				
Diagramme réseau				
Réseau final				
Michel-André Robillard	Date de changement 2025-01-06	Par MAR	2024-10-02	Projet ISPconfig
Révision: 0.0.7			Code	