Administration et gestion des réseaux

Pierre Bettens

Un grand pouvoir implique de grandes responsabilités

version *alpha*

***root* est-il Dieu ?**	**3**
***nix, les bases**	**4**
Rappel des bases réseau	**25**
Le shell	**33**
SSH - Secure shell	**44**
Monitoring et fichiers de logs	**49**
DNS domain name system	**51**
SAMBA ou l'intégration de machines MS Windows et GNU Linux	**72**
PAM, Pluggable Authentication Modules	**83**
LDAP, Lightweight directory access protocol	**86**
Serveur web	**95**
Bibliographie	**104**

Administration et gestion des réseaux

version alpha

Date de publication 12 février 2024

Remercions les auteurs des excellents livres TCP/IP de Craig Hunt(Hunt, Janvier 2000) et Administration réseau sous Linux de Olaf Kirch et Terry Dawson (Dawson *et al.*, Janvier 2005) qui ont servis de référence pour ces notes ainsi que les rédacteurs, rédactrices, traducteurs et traductrices des pages de manuel et de Wikipedia.

J'en profite pour remercier tout particulièrement toutes celles et tous ceux qui aiment bien être remerciés.

Pierre Bettens
pbettens@he2b.be
bettensp@helha.be

root est-il Dieu ?

> Un grand pouvoir implique de grandes responsabilités.
>
> *Benjamin Parker*

Les tâches d'administration sont variées, elles demandent des compétences techniques et une certaine réserve.

Dans la suite, nous appellerons la personne avec le rôle d'administration : *root*.

Les tâches techniques de *root* gagnent souvent à être centralisées, sont souvent critiques au niveau de la confidentialité et de la sécurité. C'est tâches peuvent être du support de première ligne, de l'installation de machines et de services, de la maintenance et la mise à jour desdits services, la gestion des sauvegardes (*backups*), la réalisation et le maintien de la documentation, etc.

Il n'est pas nécessaire d'être un développeur ou une développeuse pour prendre en charge l'administration des réseaux mais il est essentiel de bien connaitre *son* éditeur de texte ou de code[1] et son interprèteur de commandes. Être à l'aise avec l'écriture de scripts `bash` ou autre est un atout non négligeable.

Les actions faites par *root* ont souvent un impact sur les utilisateurices[2]. Elles peuvent les empêcher de travailler. Une bonne résistance au stress et une capacité à être *multi-tâche* aidera à répondre aux demandes. Il sera parfois nécessaire de dire non à des demandes qui ne sont pas réalisables — bien qu'elles puissent paraitre[3] naturelles à la personne demandeuse qui n'a pas du vue globale sur le système d'information ou illégales bien que techniquement faisables.

Dans son rôle d'administration, *root* a accès à des informations confidentielles. Son éthique doit être irréprochable.

Mais qui est *root* ?

Sous les systèmes **nix*, *root* est l'utilisateurice privilégié, le « super utilisateur », l'administrateurice… C'est le compte ayant le *userid* `0`.

[1] Un *éditeur de texte* est un programme qui permet d'écrire du texte (sans mise en forme) et de le sauver. Un *éditeur de code* est un programme qui permet d'écrire du texte et qui offre des services à l'utilisateurice ; coloration syntaxique, indentation, raccourcis clavier… Nous ne parlons bien sûr pas de *traitement de texte*. `notepad` est un éditeur de texte, `vim`, `notepad++` sont des éditeurs de code, `LibreOffice Writer` est un traitement de texte. Ceci étant dit, mon conseil est d'être à l'aise avec `vim` et d'oublier `nano`.

[2] Dans ces notes, l'écriture tente d'être inclusive. Un peu pour céder à la mode, un peu parce que ça « m'amuse » et un peu pour essayer d'inclure plus. J'essaierai de ne pas en abuser. Je troque généralement l'accord dit « du masculin l'emporte » contre l'accord de proximité et j'ajoute quelques nouveaux mots en évitant le point médian · qui fait peur ;-)

[3] Ces notes sont écrites en orthographe réformée.

*nix, les bases

Les utilisateurices *users* et les groupes **4**
Le cas de `sudo` 7
Choisir un bon mot de passe 8
Le coin des commandes 9
`sudo` 10
Le système de fichiers **11**
Chemins relatifs, chemins absolus 13
Les différents types de fichiers 13
Les permissions 14
Le coin des commandes 14
Les processus **16**
Le coin des commandes 17
Tâches périodiques `cron` **19**
Gestionnaire de paquet `apt` **20**
`apt` 21
Démarrage du système, `systemd` **21**
Le coin de la commande 22

Cette section rappelle les bases des systèmes **nix*. Nous supposons dans la suite que la personne lisant ces notes a déjà quelques connaissances des systèmes **nix*. Nous ne rappelons ici que ce qui nous semble nécessaire à l'administration **nix* dans le cadre de ce cours.

Nous nous concentrons sur *linux*.

Les utilisateurices *users* et les groupes

> Un système correctement administré a souvent plus de *groups* que de *users*.

Lors de l'installation, un compte administrateur ayant tous les privilèges est créé. Le compte *root*. En plus de ce compte au minimum un compte sans privilège particulier est ajouté. Nous l'appellerons *user* même si le nom peut être choisi.

La (première) bonne pratique est de se connecter au système avec un compte non privilégié — le compte *user* — et de n'utiliser les privilèges de *root* que lorsque c'est nécessaire.

root devra gérer les utilisateurs du système dans deux situation différentes :

- la première si le système doit être accessibles par plusieurs personnes, il sera nécessaires de gérer les comptes. C'est évident ;
- la seconde est pour l'installation de services sur le système. Une bonne pratique est d'associer un compte à chaque service afin que chaque service tourne avec les privilèges de ce compte… et pas ceux de *root*.

  Par exemple, le service *dns* tourne sous le compte *bind*, le serveur web *apache2* avec le compte *www-data*…

Créer un compte consiste en l'enregistrer dans le système en lui donnant un *login*, *password*, un répertoire *home*, un *shell*… tout cela se traduit par une ligne dans le fichier `/etc/passwd` et une autre dans `/etc/shadow`.

```
$ cat /etc/passwd
login:passwd:uid:gid:comment:
  home:shell
user:x:1000:1000:user,,,:
  /home/user:/bin/bash
```

- `login` le nom associé au compte. La bonne pratique est d'utiliser un *login* en minuscules, sans accents ni caractères spéciaux. La longueur est limitée à 16 caractères;
- `/etc/passwd` contenait anciennement le mot de passe hashé associé au compte. Aujourd'hui, *linux* ne laisse plus un hash en lecture et le mot de passe ne se trouve plus[4] dans `/etc/passwd`.

  Le champ du mot de passe chiffré peut être vide. Dans ce cas, aucun mot de passe n'est nécessaire pour s'authentifier avec le compte donné. Cependant, certaines applications qui lisent le fichier `/etc/passwd` peuvent décider de ne donner aucun accès si le mot de passe est vide. Si le mot de passe est un « `x` » minuscule, alors le mot de passe chiffré se trouve dans le fichier *shadow* (`man 5 shadow`) ; il doit y avoir une ligne correspondante dans le fichier shadow, sinon le compte de l'utilisateur n'est pas valide. Si le mot de passe est constitué d'une autre chaine, alors il est considéré comme un mot de passe chiffré, comme indiqué dans *crypt* (`man 3 crypt`).

  Ce champ peut aussi prendre la valeur « `*` » qui précise qu'il n'est pas possible de se connecter au compte avec `login` ou la valeur « `!` » qui empêche toute connexion au compte. Ces valeurs sont généralement utilisées pour les comptes de service. Ce sont des comptes utilisés par les services et qui ne sont pas destinés à connecter un *user*.
- `uid` est l'*user id*, l'identifiant unique de l'utilisateur. La valeur 0 est celle de *root*, les autres sont libres. Les premières valeurs sont utilisées par le système. En fonction des distributions les *uid* des comptes commencent à partir d'une certaine valeur. Pour *debian*, c'est 1000.

  Attention certaines applications se basent sur l'*uid* et pas le *login* du compte. Il s'agit donc d'agir avec prudence lorsque des machines commu-

[4] S'il s'y trouve il faut revoir la sécurité de cette machine.

niquent entre elles pour ces quelques services (par exemple `nfs`) ;

- `gid` est le *group id*, l'identifiant unique du groupe. Chaque compte est associé à un groupe principal et peut être ajouté à d'autres groupes. Le groupe principal est renseigné ici, les autres dans `/etc/group`[5] ;
- `comment` série de valeurs reprenant le nom complet du compte ainsi que diverses informations ;
- `home` répertoire *home* du compte. Chemin absolu ;
- `shell` shell associé au compte. Généralement `/bin/bash`.

  Cette valeur peut être positionnée à `/bin/false` pour désactiver le compte ou pour empêche un login au compte.

Le fichier `/etc/shadow` est un fichier qui contient les informations cachées concernant les mots de passe des comptes et leurs dates de validité.

```
# cat /etc/shadow
```

```
user:$6$0--cut--Z17309:0:99999:7:::
```

Ce fichier ne doit pas être accessible en lecture par les utilisateurices normaux afin de maintenir la sécurité des mots de passe, en particuliers pour prévenir les attaques par dictionnaires.

Chaque ligne de ce fichier contient 9 champs, séparés par des deux-points (« `:` »), dans l'ordre suivant :

- `login` le login — existant — du compte concerné ;
- `password` le mot de passe *hashé*.

  Si le champ du mot de passe contient une chaine qui ne peut pas être un résultat valable de crypt(`man 3 crypt`), par exemple si elle contient les caractères `!` ou `*`, alors l'utilisateur ou l'utilisatrice ne pourra pas utiliser son mot de passe UNIX pour se connecter (mais il se peut que le compte puisse se connecter au système par d'autres moyens).

  L'algorithme de chiffrement utilisé est renseigné dans le fichier `/etc/login.defs`. Ce devrait être au moins `SHA512`

  ```
  ENCRYPT_METHOD SHA512
  ```

- `date du dernier changement de mot de passe` la date du dernier changement de mot de passe, exprimée en nombre de jours depuis le 1er janvier 1970.

  Quand cette valeur vaut `0` un changement de mot de passe est requis à la prochaine connexion.

  Quand la valeur est absente (champ vide), les fonctionnalités de vieillissement de mot de passe sont désactivées.

- `âge minimum du mot de passe` l'âge minimum du mot de passe est la durée (en jour) que l'utilisateur devra attendre avant de pouvoir le changer de nouveau.

  Un champ vide ou une valeur de 0 signifie qu'il n'y a pas d'âge minimum pour le mot de passe.

[5] Les lignes sont de la forme `group:password:gid:users_list`. La commande `groups` donne la liste des groupe d'un *user*. Un *grep* pourrait faire l'affaire `grep user /etc/group`.

- `âge maximum du mot de passe` l'âge maximum du mot de passe est la durée (en jour) après laquelle l'utilisateur devra changer son mot de passe.

  Une fois cette durée écoulée, le mot de passe restera valable. Il sera demandé à l'utilisateur de le changer la prochaine fois qu'il se connectera. Un champ vide signifie qu'il n'y a pour le mot de passe aucune limite d'âge, aucune période d'avertissement d'expiration et aucune période d'inactivité (voir ci-dessous).

- `période d'avertissement d'expiration du mot de passe` la durée (en jour) pendant laquelle l'utilisateur sera averti avant que le mot de passe n'expire (voir l'âge maximum du mot de passe ci-dessus).

  Un champ vide ou une valeur de 0 signifie qu'il n'y aura pas de période d'avertissement d'expiration du mot de passe.

- `période d'inactivité du mot de passe` la durée (en jour) pendant laquelle le mot de passe sera quand même accepté après son expiration (voir l'âge maximum du mot de passe ci-dessus). L'utilisateur devra mettre à jour son mot de passe à la prochaine connexion.

  Après expiration du mot de passe suivie de la période d'expiration, plus aucune connexion n'est possible en utilisant le mot de passe de l'utilisateur. L'utilisateur doit contacter son administrateur. Un champ vide signifie qu'aucune période d'inactivité n'est imposée. date de fin de validité du compte La date d'expiration du compte, exprimé en nombre de jours depuis le 1er janvier 1970.

  Un champ vide signifie que le compte n'expirera jamais.

  La valeur 0 ne doit pas être utilisée puisqu'elle peut être interprétée soit comme un compte sans expiration, soit comme ayant expiré le 1er janvier 1970.

- le dernier champ est réservé pour une utilisation future.

Outre ces deux entrées dans les fichiers `/etc/passwd` et `/etc/shadow` la création d'un compte entraine la copie des fichiers contenu dans `/etc/skel` dans le répertoire *home* du compte. C'est là que *root* peut paramétrer certains fichiers de configuration avant la création d'un compte. Par défaut, ce répertoire contient :

```
alice@harmony:~$ tree -a /etc/skel
/etc/skel
├── .bash_logout
├── .bashrc
└── .profile

0 directories, 3 files
```

Un manière simple de désactiver un compte est de changer son *shell* dans le fichier `/etc/passwd` et le remplacer par `/bin/false` par exemple.

Le cas de `sudo`

Certaines tâches d'administration ne peuvent être réalisées que par *root*. Pour ces tâches :

- il est possible de changer d'identité et devenir *root* grâce à la commande `su` ou;
- il est également possible d'exécuter la commande en tant que *root* grâce à la commande `sudo`.

Pour devenir *root* — avec `su` — il est nécessaire de fournir le mot de passe *root* tandis que pour exécuter une commande en tant que *root* — avec `sudo` — il faut fournir le mot de passe de l'utilisateur et avoir été préalablement autorisé par *root* à exécuter cette commande.

Les actions faites avec `sudo` sont loguées.

`sudo` est un utilitaire qu'il est nécessaire d'installer et configurer[6].

Choisir un bon mot de passe

Le choix d'un bon mot de passe est primordial pour les mots de passe sensibles comme les mots de passe des comptes administrateurs et donc, *root* mais également pour les mots de passe des comptes utilisateurs sans privilège particulier. À ce sujet, il sera possible de mettre en place une *politique de mot de passe*.

En préambule, respecter les règles de bonne utilisation des mots de passe est contraignant et difficile. C'est cependant un des éléments principaux de la sécurité informatique.

Voici quelques règles habituelles :

- un mot de passe devrait être spécifique à un service informatique et ne devrait pas être réutilisé : un service = un mot de passe spécifique et unique;
- un mot de passe est personnel et ne peut être donné à personne;
- un bon mot de passe est facile à retenir pour l'utilisateur qui le définit et difficile à trouver par une machine ou quelqu'un d'autre;
- un mot de passe ne devrait pas être basé sur des informations personnelle qui peuvent être facilement identifiées ou devinées;
- une bonne manière de faire est qu'il soit long, et ne puisse pas figurer dans une liste ou un dictionnaire;
- pour définir un bon mot de passe, par exemple accoler des mots en incorporant des majuscules, des minuscules, des chiffres et des caractères spéciaux (car un mot de passe ne peut souvent pas contenir d'espace). Ou encore prendre les initiales des mots d'une phrase. C'est mieux également s'il mélange les langues;

  Par exemple si je pense aux trois mots « table », « chaise » et « manger », un bon mot de passe pourrait être `tableCHAISEeat723`.

  De part sa longueur ce mot de passe sera résistant — à l'heure où j'écris — à une force brute caractère par caractère mais aussi à une recherche mot par mot puisqu'il utilise deux langues et des chiffres.
- pour un mot de passe de service, il est inutile qu'il soit retenu et il est tout à fait envisageable de le stocker dans un gestionnaire de mots de passe.

Nous verrons avec `PAM` qui est possible de définir des règles sur le format des mots de passe.

[6] Certaines distributions l'installe par défaut, d'autres non.

Le coin des commandes

adduser et addgroup adduser et addgroup ajoutent des comptes utilisateurs ou des groupes au système en fonction des options et du fichier de configuration /etc/adduser.conf. Ces commandes choisissent un *uid* et *gid* conformes à la charte *debian*, crée un répertoire personnel configuré en fonction du contenu de /etc/skel, d'exécuter un script sur mesure.

```
adduser <username>
```

Ajoute un compte utilisateur normal en lui associant le premier *uid* disponible et un groupe — créé si nécessaire — du même nom que le compte utilisateur.

- Attribue un masque de 002 au compte.
- Crée un répertoire personnel. Par défaut tous les fichiers créés dans le répertoire personnel du compte utilisateur auront le bon groupe.
- Copie les fichiers *skel* dans le répertoire personnel du compte utilisateur.
- Si le fichier /usr/local/sbin/adduser.local existe, il sera exécuté.

```
adduser --system <username>
```

Ajoute un compte utilisateur système.

- Par défaut les comptes utilisateurs systèmes sont placés dans le groupe nogroup.
- Un répertoire personnel est également créé.
- Le shell par défaut sera /usr/sbin/nologin.

```
adduser --group <groupname>
addgroup <groupname>
addgroup --system <groupname>
```

Ces trois commandes ajoutent un groupe au système sans utilisateur. La dernière ajoute un groupe système qui diffère d'un groupe normal par son *gid* qui n'est pas dans le même intervalle.

```
adduser <username> <groupname>
```

Ajoute le compte utilisateur *username* au groupe *groupname*.

useradd et usergroup Ces commandes sont les commandes bas niveau associées aux commandes précédentes. Elles créent les compte en fonction des paramètre donnés en option sur la ligne de commande.

userdel et deluser Comme pour les autres commandes userdel est la commande de bas niveau pour laquelle il est nécessaire de passer en arguments les options désirées tandis que deluser est de plus haut niveau.

userdel se contente de supprimer les entrées de /etc/passwd et etc/shadow, elle ne supprime par exemple pas le groupe principal associé au compte utilisateur. Par contre, elle ne supprimera pas le compte s'il a des processus actifs.

deluser et delgroup retirent des comptes utilisateur et des groupes du système

suivant les options et les informations contenues dans `/etc/deluser.conf` et `/etc/adduser.conf`.

```
deluser <username>
deluser --remove-home \
  --remove-all-files <username>
```

Retire un compte utilisateur normal.

- Par défaut le répertoire personnel n'est pas supprimé, c'est l'option `--remove-home` qui s'en charge.
- L'option `--remove-all-files` recherche tous les fichiers que *username* possède et les supprime.
- Il est possible de faire une sauvegarde des fichiers du compte utilisateurs *via* l'option `--backup`.
- Si le fichier `/usr/local/sbin/deluser.local` existe, il sera exécuté après la suppression du compte. Ceci permet un nettoyage supplémentaire.

```
deluser --group <groupname>
delgroup <groupname>
```

Retire un groupe du système.

```
deluser <username> <groupname>
```

Retire le compte utilisateur du groupe.

`chage` La commande `chage` modifie le nombre de jour entre les changements de mot de passe et la date du dernier changement.

```
chage -M 30
```

Cette commande force le changement du mot de passe chaque mois.

`su` La commande `su` — pour *substitute* — permet d'exécuter une commande sous le compte d'un autre compte utilisateur.

Sans argument, `su` exécute un *shell* en *root*.

L'option `-` (ou `-l` ou `--login`) lance le *shell* comme un *shell* de login:

- efface les variables d'environnement exceptée `TERM`;
- initialises les variables d'environnement `HOME`, `SHELL`, `USER`, `LOGNAME` et `PATH`[7];
- change de répertoire vers le répertoire cible;
- place le premier argument (`argv[0]`) du shell à `-` pour en faire un shell de login.

`su` utilise PAM.

```
su -
su alice -c "ls -il"
```

`sudo`

La commande `sudo` permet d'exécuter une commande en tant que *root* ou d'une autre *user*. La commande est loguée.

[7] Ce point est source d'erreurs. Une exécution de `su` — sans le `-` donc — ne charge pas le `PATH` de *root* mais conserve celui de l'utilisateurice sans privilège. Les répertoires `/sbin` et `/usr/sbin` n'en font par exemple pas partie.

Le système de fichiers

Une machine **nix* n'a qu'un seul système de fichier (*filesystem*) dont la racine se note `/`. Ce *filesystem* représente comment on accède aux « informations » stockées sur un « support ».

- *l'information* peut bien sûr être un fichier ou un programme mais elle peut aussi être un *pseudo-fichier* faisant le lien avec un composant matériel (*hardware*).
- *le support* sera une partition d'un disque dur bien sûr mais également un accès à un partage distant accessible par le réseau. Il pourra aussi être un pseudo-fichier accédant au matériel.

La structure du *filesystem* *nix est standardisée par le « Filesystem Hierarchy Standard »Wikipedia (s.d.)· Internet (s.d.-b) — même si les différentes distributions ne respectent pas le standard à la lettre, les grandes lignes le sont.

Le groupe en charge de l'harmonisation de *filesystem* à choisi de distinguer :

- les fichiers **partageables** entre plusieurs machines de ceux qui ne le sont pas. Les pages de manuel peuvent par exemple être partagées;
- les fichiers **variables** et ceux qui le sont peu. Les fichiers variables ont un contenu et une taille qui varie fortement pendant la vie du programme qui les utilise et donc du système. Par exemple, les mails entrants et sortant.

En voici un résumé :

Répertoire	*Description*
`/bin`	Les commandes de base nécessaires au démarrage et à l'utilisation d'un système minimaliste (`ls`, `cat`…) exceptées les commandes *root* qui se trouvent dans `/sbin`. *bin pour binaire.*
`/boot`	Les fichiers nécessaires au démarrage (par ex. le noyau et *grub*). *boot pour… boot*
`/dev`	Les *pseudo-fichiers* correspondant à un périphérique *hardware*. *dev pour devices*
`/etc`	Fichiers de configuration. Contient souvent un répertoire pour le programme concerné (par ex. `/etc/apache2/`). *etc pour editable text configuration.*
`/home`	Répertoires des *users* du système (par ex. `/home/alice`) *home pour « qu'il fait bon chez moi ».*
`/lib`	Bibliothèques logicielles (*libraries*) nécessaires aux binaires de `/bin` et `sbin`. *lib pour libraries.*
`/mnt`	Point de montage pour les systèmes de fichiers temporaires. *mnt pour mount.*

Répertoire	*Description*
`/media`	Point de montage pour les médias amovibles (anciennement les CD-ROM, aujourd'hui les clés USB et demain…). *media pour medias*
`/opt`	Logiciels optionnels, ce sont ceux qui ne sont pas proposés par la distribution et installés pour tous les *users*. Il est préférable d'utiliser `/opt` à `/usr/local`. *opt pour optionals*.
`/proc`	Système de fichiers virtuel pour les processus. *proc pour processes*
`/root`	Répertoire *home* de *root*.
`/sbin`	Binaires pour *root*. *sbin pour system binaries*.
`/srv`	Données pour les services hébergés (contenu web statique, base de donnée…). *srv pour services*.
`/tmp`	Fichiers temporaires. Le répertoire est vide au démarrage. *tmp pour temporary*.
`/usr`	Arborescence semblable à la racine pour les fichiers et répertoires qui ne sont pas nécessaires au fonctionnement minimals du système (par ex. `/usr/bin`, `/usr/lib`, `/usr/sbin`, `/usr/share`, `/usr/include`…) *usr pour unix system resources*
`/var`	Destiné à recevoir des fichiers variables divers. *var pour variables*.
`/var/cache`	Pour différents cache (par ex. *bind*, *apt*).
`/var/lock`	Fichiers de verrouillage.
`/var/mail`	Boites mails des utilisateurs.
`/var/spool`	Données en attentes de traitement (par ex. pour l'impression, les mails, les tâches planifiées…).

Remarques :

- Certains binaires se trouvant dans `/sbin` peuvent être exécutés par un *user* sans privilège tant que cet *user* ne demande pas une action à laquelle il ou elle n'a pas droit. Par contre `/sbin` ne se trouve pas dans son `PATH`.
- Historiquement un petit disque — rapide — était réservé aux fichiers essentiels — se trouvant dans `/bin`, `/sbin`… — tandis que les autres étaient placés dans `/usr` qui pouvait donc se trouver sur un autre disque.
- Les répertoires `/usr/src` et `/usr/include` sont plutôt destinés à recevoir tout ce qui est nécessaires à la compilation C ou C++ des différents logiciels se trouvant sur la machine.

Chemins relatifs, chemins absolus

Les noms de fichiers commençant par `/` sont des noms de fichiers **absolus** c'est-à-dire faisant référence à la racine du *filesystem*. Le nom est indépendant du répertoire courant.

Il existe un raccourci représentant le répertoire *home* de l'utilisateurice : `~`. Pour l'utilisatrice *alice*, il s'agit de `/home/alice`.

Par exemple :

```
/etc/apache2/apache2.conf
~/bin/yascript.sh
```

Les noms de fichiers **relatifs** sont relatifs au répertoire courant, ils s'expriment sans la référence à la racine `\`.

Les raccourcis `.` et `..` représentent respectivement le répertoire courant et le répertoire parent.

Par exemple si le répertoire courant est `/home/alice` :

```
../../etc/apache2/apache2.conf
bin/yascript.sh
```

Les différents types de fichiers

Un fichier est désigné par un **nom** bien sûr et possède un ***inode*** unique. L'*inode* d'un fichier contient : le type de fichier, les droits d'accès, le nombre de liens physiques, un *uid* du propriétaire, un *gid*, la taille du fichier, les dates d'accès, de modifications, les connexions et l'adresse du fichier.

Les différents types de fichiers sont :

- fichier ordinaire. Quasi tout ce qui peut être enregistré est un fichier;
- répertoire (*directory*). C'est un fichier contenant une table associant un nom à un *inode*. Les noms et les *inodes* des fichiers qu'il contient;
- fichier de périphérique. Les *peusdo-fichiers* faisant le lien vers du matériel (disque, terminal, sortie parallèle, sortie série, sortie usb…). Ils se trouvent dans `/dev`;
- lien physique et lien symbolique :
  - un lien physique est un fichier contenant l'*inode* du fichier qu'il référence. C'est un nom supplémentaire pour le même inode;

    À chaque ajout d'un lien physique le nombre de référence vers le fichier augmente. À chaque suppression, il diminue. Lorsque le nombre de référence passe de 1 à 0, le fichier est supprimé.
  - un lien symbolique montre le chemin vers le fichier pointé. C'est un nouveau fichier — avec un nouvel *inode* — contenant l'*inode* du fichier vers lequel le lien pointe.

```
$ echo "foo" > file
$ ln file fileln
$ ln -s file filelns
$ ls -il
total 8
262155 -rw-r--r-- 2 pbt pbt 5 fév 18 22:52 file
262155 -rw-r--r-- 2 pbt pbt 5 fév 18 22:52 fileln
262158 lrwxrwxrwx 1 pbt pbt 4 fév 18 22:53 filelns -> file
```

- tube nommé (*named pipe*). Un tube nommé est un tube… nommé. Comme la commande *pipe* « `|` », un tube permet de relier la sortie d'un processus à l'entrée d'un autre. Un pipe nommé permet à un processus d'écrire dans un «_fichier sans fin ». Pendant ce temps, un autre processus peut lire dans ce « fichier sans fin ».

Les permissions

`ugo` pour *user*, *group* et *other*. Un fichier, quel que soit son type a des permission pour son propriétaire, son groupe et « les autres ». Les autres étant tous les *users* n'appartenant pas au groupe.

Pour chacun d'entre eux, les droits peuvent être `rwx` pour *read*, *write* et *execute*. Les droits d'un fichier se présentent comme suit :

```
-rwxr---wx 1 alice yagroup  46K fév 11 16:11 filename.pdf
```

- `r` permet de lire un fichier ou de voir le contenu d'un répertoire;
- `w` permet d'écrire dans un fichier ou d'écrire dans un répertoire. Écrire dans un répertoire, c'est ajouter, effacer, renommer un fichier;
- `x` indique que le fichier est exécutable (il peut s'agir d'un binaire ou d'un script) ou que le répertoire est « traversable ».

Le coin des commandes

`ls` `ls` donne la liste des fichiers d'un répertoire. Voici quelques options :

- `-l`, format long donne les droits, le nombre de liens vers le fichier, le propriétaire, le groupe, la taille, la date de dernier accès et le nom;
- `-i`, affiche le numéro d'*inode* du fichier;

`cd` `cd` (*change directory*) change le répertoire courant.

```
cd /home/alice
cd ..
cd ~/bin
cd -
```

L'option `-` change le répertoire vers le répertoire précédent (pas le parent).

`chmod` `chmod` permet de changer le propriétaire et le groupe d'un fichier ou d'un répertoire, le *mode*. Le mode peut être représenté de manière symbolique ou octale.

- représentation symbolique `[ugoa...][-+=][perms]` où *perms* est 0 ou plus parmi les lettre `rwxXst`. Plusieurs modes symboliques peuvent être donnés, séparés par des virgules;
- représentation octale est composée de maximum 4 chiffres octaux. Le premier permet de placer le *user id* ou le *group id* ou le *sticky bit*, le second pour le *user* `u`, le troisième pour le *group* `g` et le quatrième pour *other* `o`.

  L'usage courant se compose des trois derniers.

```
chmod u+x, go-w foo
chmod 510 bar
```

`chgrp` Change le groupe de chaque fichier.

```
chgrp newgroup file
```

`du` **et** `df` Ces deux commandes donnent des informations sur l'espace disque. `du` (*disk usage*) donne l'espace utilisé pour un répertoire tandis que `df` (*disk free*) donne l'espace disponible sur la partition concernée.

Option `-h` donne les résultats de manière « *human readable* ».

`u|mount` `mount` et `umount` permettent d'ajouter / retirer un support au système. Pour qu'un *filesystem* soit accessible il doit être monté sur un répertoire.

Les montages habituels d'un système sont ceux réalisés au *boot* du système et ceux lorsque l'utilisateurice insère une clé USB. Le premier montage est fait automatiquement grâce aux renseignements se trouvant dans le fichier `/etc/fstab` tandis que l'autre est actuellement pris en charge par un « auto-mount » dès que l'insertion de la clé est faite. En tant que *root* il sera parfois nécessaire de manipuler les différentes partitions.

```
mount
mount /dev/sdb1 /mnt/backup
```

- la première commande (sans paramètre) montre l'état des différents montages;
- le seconde monte la première partition du second disque scsi dans le répertoire `/mnt/backup` (qui doit exister au préalable).

`touch` `touch` « touche » le fichier. L'effet est de changer la date d'accès au fichier ou de créer un fichier vide s'il n'existait pas.

`mkfifo` `mkfifo` crée un pipe nommé (*named pipe*).

Les processus

Un processus est un programme en cours d'exécution. Un programme pouvant être exécuté plusieurs fois en même temps, il est possible d'avoir plusieurs instances du même programme au même moment.

Un processus se caractérise par :

- un *pid*, identifiant de processus (*process id*) ;
- un *ppid*, identifiant du processus parent (*parent process id*) ;
- un *uid*, l'identifiant de *user* qui a lancé le processus ;
- un *guid*, l'identifiant du groupe du *user* qui a lancé le processus ;
- un *euid*, l'identifiant de *user* « effectif » (*effective user id*) qui a lancé le processus_;
- un *egid*, l'identifiant du groupe du *user* qui a lancé le processus ;
- un *état*, état (*state*) du processus (voir ci-dessous);
- une *priorité*

gid est un alias de *egid*. *euid* est un alias de *uid*.

Un processus n'est pas toujours en cours d'éxécution (*running*) puisqu'il y a plus de processus que de CPU. La plupart des processus sont soit en cours d'exécution (***run***), soit prêt (***ready***), soit en attente (***wait***).

À sa création, un processus est placé dans le statut prêt (*ready*) et attend d'être choisi par le *scheduler*.

Un processus *running* devient « en attente » (*waiting*) lorsqu'il attend des ressources qui ne sont pas encore disponibles ; entrées-sorties ou tout autre évènement. Il se met en seul en attente ou le *kernel* s'en charge. Lorsque les ressources attendues seront disponibles, le *scheduler* mettra le processus dans l'état prêt (*ready*)._

Voici les différents états d'un processus tels que présentés dans la page de manuel :

s (*state*)	**État du processus**
`R`	*running* or *runnable*, le processus est en cours d'exécution (*running*) ou est prêt à l'être (*runnable*) (il ne lui manque que le CPU).
`S`	*interruptible sleep*, le processus est en attente (*wait*) et peut être interrompu.
`D`	*uninterruptible sleep*, le processus est en attente (*wait*) et ne peut pas être interrompu (probablement en I/O).
`I`	*idle kernel thread*, le processus est *idle* ce qui signifie qu'il est à la fois *uninterruptible sleep* et *no load*. Ce processus est en attente de travail et ne participe pas au calcul de la charge du système [8].

s (*state*)	**État du processus**
`T`	*stopped by job control signal*, le processus est stoppé (par un signal `SIGSTOP`). Il redeviendra prêt lorsqu'il recevra un signal `SIGCONT`.
`t`	*stopped by debugger*, le processus est stoppé par un signal reçu d'un débogueur (*debugger*).
`X`	*dead*, le processus est terminé. Il a terminé son `run()`. Cet état n'apparait normalement pas car un processus terminé est détruit et n'apparait plus.
`Z`	*zombie*, le processus est terminé mais n'a pas été récupéré par son parent ni par *init*.

Le processus parent de tous les processus est **init** de *pid* 1.

Un processus peut être lancé en tâche de fond (*background*) en entrant la commande suivie d'une esperluette `&`.

Un processus peut-être stoppé en lui envoyant le signal `SIGSTOP` au moyen, par exemple, d'un `Ctrl-Z`. Il peut alors être relancé — continué — en le plaçant en tâche de fond (*background*) *via* `bg <job number>` par exemple.

Le coin des commandes

`ps` `ps` liste les processus du système. Sans option[10], liste les processus associés à la console (`TTY`) courant.

- `u` liste les processus appartenant au compte utilisateur courant;
- `a` liste tous les processus associés à un terminal;
- `ax`, `-e`, `-A` liste tous les processus;
- `faux` liste tout des processus sans restreindre aux processus de *user* (`a`), sans restreindre aux processus sans TTY (`x`), les sélectionne par *uid* (`u`) et les affiche dans le format *full* (`f`). Ce format liste les processus sous forme d'arbre.

`kill` `kill <pid>` envoie un signal au processus *pid*.

Même si les signaux les plus courants sont les signaux de mort ou de demandes de mort, il en existe d'autres. La liste s'obtient par l'option `-L` ou `-l`.

```
kill -L
```

[8] Voir ce kernel commit[9] (*consulté le 10 fév. 2021*)

[10] Cette commande est un peu particulière car elle accepte le « style UNIX » où les options peuvent être groupées et précédées d'un tiret (*dash*) « - », le « style BSD » où les options peuvent être groupées et ne doivent pas être précédées d'un tiret et le « style GNU » où les options sont au format long et sont précédées de deux tirets.

Par défaut, *kill* envoie le signal `TERM` qui demande au processus de se terminer ; une demande de mort. Pour insister et envoyer le signal de mort `SIGKILL`, il faut le préciser:

```
kill -SIGKILL <pid>
kill -KILL <pid>
kill -9 <pid>
```

Si vous avez stoppé un processus *via* `Ctrl-Z` (ou en lui envoyant le signal `STOP` par `kill -STOP <ipd>`), vous pouvez lui envoyez le signal `CONT` par un

```
kill -CONT <pid>
```

Le signal `SIGHUP` est aussi intéressant car il demande à certaines applications de relire leur fichier de configuration.

Attention, le *pid* peut valoir `-1`, dans ce cas, il signifie : tous les processus exceptés *init* et le processus *kill* lancé. En ce sens, la commande suivante est déconseillée :

```
kill -9 -1
```

`top` et `htop` `top` et sa version *++* `htop` montre la liste des processus.
`q` pour quitter.

`[h]top` supporte la navigation avec les touches `Up`, `Down`, `Left`, `Right`, `PgUp`, `PgDn` et les commandes suivantes (extrait, cfr. *man htop*) :

- `Space` *tag* ou *untag* un processus;
- `U` *untag* tous les processus;
- `l` affiche les fichiers ouvert par ce processus (nécessite `lsof`);
- `t` *tree view* vue en arbre;
- `k` envoie un signal sélectionnable dans un menu;
- `u` affiche uniquement les processus d'un *user*;
- `M` trie par usage mémoire;
- `P` trie par usage processeur;
- `F` sélectionne une ligne et suit (*follow*) le processus si la liste est retriée par exemple;

Par défaut, `htop` montre

- `PID` le *process id* ;
- `USER` le *user*;
- `PRI` la priorité du processus (habituellement, la valeur de *nice* augmentée de 20);
- `NI` la valeur de *nice*,
- `VIRT` la taille de la mémoire virtuelle (*virtual*) du processus;
- `RES` la taille résident en mémoire du processus (text + data + stack);
- `SHR` la taille des pages partagées en mémoire (*shared*);
- `S` le statut de processus *state*;
- `CPU%` le pourcentage CPU utilisé;
- `MEM%` le pourcentage mémoire;
- `TIME+` le temp en click horloge utilisé par le processus;
- `Command` la commande complète du processus.

`pstree` `pstree` montre des processus sous forme d'arbre.

```
pstree
pstree <user>
pstree -ph
pstree <user> -ph
pstree <pid> -ph
```

- `-p` montre les *pid* et désactive la vue compacte;
- `-h` surligne le procesus courant et ses parents

`pidof` `pidof` donne le *pid* — ou les s'il y a plusieurs instance du même programme — du processus dont le nom est donné.

```
pidof init
pidof bash
```

Tâches périodiques `cron`

Les systèmes linux disposent du programme `cron` capable d'exécuter un tâche à un moment donné. Chaque utilisateur dispose d'un fichier *crontab* et d'une commande associée permettant d' 'y accéder.

```
crontab -e
```

Cette commande permet d'éditer le fichier *cron* de l'utilisateur. Ce fichier a l'allure suivante:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Pour qu'une tâche particulière soit exécutée à intervalle régulier — par exemple tous les jours pendant la nuit pour faire un backup — le script correspondant à cette tâche doit se trouver dans ce fichier.

Au fichier *crontab* de chaque compte sont associés des fichiers *crontab hourly*, *daily* et *weekly*. Ces fichiers sont lancés et

référencés dans `/etc/crontab`.

Certains programmes placent leur fichier *cron* dans `/etc/cron.d`.

Pour empêcher ou permettre des comptes à utiliser *cron*, *root* peut créer des fichiers `/etc/cron.allow` et `/etc/cron.deny` contenant les comptes utilisateurices pouvant utiliser ou non *cron*. Généralement, tous les comptes peuvent utiliser *cron* et ces fichiers ne sont pas créés.

Remarque : `cron` n'est pas le seul programme permettant d'effectuer des tâches au temps `t`, *systemd* et ses *timers* aussi bien que ce soit beaucoup moins répandu.

Gestionnaire de paquet `apt`

`apt` (*advanced package tool*) est l'outil de gestion des paquets *debian*.

Une distribution linux est plus qu'un simple noyau linux, c'est un ensemble cohérent de logiciels qui sont « distribués » par une « entité ». Cette entité peut-être une entreprise ou une communauté. Une distribution linux rassemble les logiciels en un ensemble **cohérent** mais également **stable** et offre un système de maintenance de ces logiciels.

Chaque distribution a ses particularités : l'usage (bureautique, serveur…), le matériel sur lequel l'installer, la facilité d'utilisation, les choix prédéfinis, le support… Il faudra donc à un moment **choisir** sa distribution linux. Pour le cours, le choix s'est porté sur ***debian***.

En fonction de la distribution choisie, le système de gestion des logiciels — les « paquets » — diffère. Les plus connus sont les systèmes basés sur les paquets `deb` et `rpm`.

Un paquet `deb` est une archive au format `ar` contenant elle-même deux archives ; `control.tar.gz` et `data.tar.gz` et un fichier de version (`debian-binary`). Les paquets debian se manipulent à l'aide de la commande `dpkg` pour une gestion bas niveau paquet par paquet. La gestion quotidienne de son système se fait quant-à elle grâce à `apt`[11].

Les paquets *debian* sont disponibles sur plusieurs sites et il est également possible d'installer et maintenir un **miroir** de l'un de ces sites. La première chose à faire est de configurer le miroir utilisé. Cela se fait dans le fichier `/etc/apt/sources.list` qui pourrait avoir l'allure suivante[12] :

```
deb http://ftp.be.debian.org/\
 debian/bookworm main contrib non-free
deb-src http://ftp.be.debian.org/\
 debian/bookworm main contrib non-free

deb http://security.debian.org/\
 bookworm/updates main
deb-src http://security.debian.org/\
 bookworm/updates main
```

[11] Historiquement, la gestion des paquets debian se faisait à l'aide des commandes `apt-get` et `apt-cache`. Ensuite la commande `aptitude` a été conseillée. Aujourd'hui, la commande `apt` suffit.

[12] Pour une configuration en Belgique, choisir un miroir proche. Le miroir de son fournisseur d'accès est un bon choix. Si beaucoup de machines debian sont installées dans son entreprise maintenir un miroir local est sans doute une bonne idée.

- *bookworm* est le nom de la *release* stable à l'heure de la révision de ces notes[13].

- *main contrib non-free* seul *main* est nécessaire à l'installation du système, les autres valeurs peuvent être ajoutées pour accéder aux paquets *contrib* ou *non-free*. Les paquets *non-free* peuvent être utile pour un *driver* propriétaire particulier par exemple.

`apt`

- `apt update` met à jour la liste des paquets disponibles ainsi que leur version en local ;

- `apt list --upgradable` liste les paquets qui peuvent être mis à jour ;

- `apt upgrade` met à jour le système en téléchargeant les paquets et en les installant sur le système ;

- `apt search <pattern>` fait une recherche dans la liste des paquets disponibles à la recherche d'un paquet correspondant au *pattern* ;

```
# apt search bind9
En train de trier... Fait
Recherche en texte intégral... Fait
bind9/stable,now 1:9.11.5[cut] \
    amd64  [installé]
  Serveur de noms de domaines internet

bind9-doc/stable 1:9.11.5[cut] all
  documentation de BIND
[cut]
```

- `apt install <paquet>` installe le paquet *paquet* et les paquets dont il dépend ;
  - `--reinstall` cette option demande de faire une réinstallation du paquet ;

- `apt remove` désinstalle un paquet. Ne désinstalle pas les dépendances ;

- `apt autoremove` désinstalle les paquets que ne sont plus nécessaires ;

Remarque : Lors de l'installation d'un paquet, toujours faire un *update* avant une installation.

En effet, la procédure d'installation va tenter d'aller chercher le paquet dans la version renseignée dans la liste des paquets disponibles **locale**. Si le paquet a été mis à jour sur le miroir, `apt` tentera de télécharger un fichier qui n'existe plus.

Démarrage du système, `systemd`

Au démarrage du système, les étapes suivantes sont exécutées :

- à la mise sous tension, le système charge le *moniteur*, anciennement `BIOS` et actuellement `uefi` qui fait

[13] Il est conseillé de renseigner le nom de la *release* plutôt que `stable` pour éviter un saut de version inopiné lors d'une mise à jour du système. Chez debian, elles se nomment : Hamm, Slink, Potato, Woody, Sarge, Etch, Lenny, Squeeze, Wheezy, Jessie, Stretch, Buster, Bullseye et Bookworm.

la vérifications matérielles (CPU, mémoires, périphériques…) et initie le *bootstrapping*.

Ce programme peut être interrompu par configurer le *boot* du système et certaines configurations matérielles ;

uefi au contraire de *BIOS* peut résider sur une partition du disque, le firmware *uefi* peut donc lire et charger le code se trouvant sur une petite partition. C'est là que se trouve le code *uefi* de *grub*.

- le moniteur est configuré pour chercher du code sur certains périphériques dans un certains ordre (un disque dur, une clé USB…) ;
- chargement du code `grub2`[14].

  Il est possible d'interrompre *grub* et de passer des options au noyau sur lequel le système va booter.
- si c'est le choix de linux qui est fait dans *grub*, chargement du noyau sélectionné — éventuellement en deux étapes — et montage de la partition système `/` ;
- chargement de *systemd*, le « super dæmon » dont la responsabilité est de lancer tous les services prévus dans l'ordre qui va bien. C'est-a-dire en gérant les dépendances et l'ordre suggéré par *root*.

  Dans un système fonctionnant avec *systemd*, le processus de *pid* `0` appelé `init` est `systemd`.

  Dans un système fonctionnant avec *Sys V*, le processus de *pid* `0` appelé `init` est (était) le script exécutant le fichier `/etc/inittab` et lançant ensuite l'exécution des scripts se trouvant dans `/etc/rci.d` où `i` est le *runlevel* choisi au *boot* de la machine. *Sys V* a été abandonné au profit de *systemd* par debian avec *Jessie*[15].

systemd est un gestionnaire de système et de services pour linux. *systemd* est compatible avec les scripts d'initialisation *Sys V* et remplace *sysvinit*. Un des principaux avantages avancés pour *systemd* est qu'il permet la parallélisation.

Il s'organise par **unités** (*unit*) ; les services (*.services*), les points de montage (*.mount*), les périphériques (*.device*)… Ces unités sont définies dans un fichier. Ces fichiers se trouvent dans `/etc/systemd/`, `/lib/systemd/system/`… (cfr. `man systemd.unit`).

Les unités sont rassemblées pour former des **cibles** (*target*) ayant un sens ; par exemple la cible `graphical.target` ou `sound.target`. La configuration de ces cibles comprend ce dont elles ont besoin pour pouvoir être exécutées.

Au niveau de la sécurité des processus, *systemd* place chaque service dans un groupe de contrôle (*cgroup*) dédié au service. Ceci permet une bonne / meilleure isolation du système.

Le coin de la commande

`systemctl systemctl` est la commande principale pour contrôler et gérer l'état du système, lancer, stopper des services.

[14] Les *bootloader* linux sont, dans l'ordre d'apparition et de disparition : *LILO*, *Grub* et *Grub2*.
[15] Ou *Debian 8*.

```
# systemctl [options] command [unit]
```

- `status [pattern | pid]` sans option, affiche l'état du système sinon affiche l'état de la cible, de l'unité… ;

Exemples sans les retours de commandes

```
$ systemctl status
$ systemctl status bind9.service
$ systemctl status 9823
$ systemctl /dev/sda
```

Exemple de sortie du status de bind9

```
$ systemctl status bind9
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset:
           enabled)
   Active: active (running) since Tue 2021-02-23 08:02:16 CET; 3 days ago
     Docs: man:named(8)
 Main PID: 6792 (named)
    Tasks: 7 (limit: 4915)
   Memory: 16.4M
   CGroup: /system.slice/bind9.service
           └─6792 /usr/sbin/named -u bind
```

- `--failed` affiche les services qui ont échoués ;
- `list-units [pattern]` affiche les unités que *systemd* a actuellement en mémoire [16] ;

```
$ systemctl list-units \
    --state=running
```

- `list-unit-files` affiche les fichiers correspondants aux différents services ;
- `start [pattern]`
  `stop [pattern]`
  `reload [pattern]`
  `restart [pattern]`
  `reload-or-restart [pattern]` lance, stoppe, demande de relire le fichier de configuration (*reload*), stoppe et relance ou reloade ou restart le service correspondant au *pattern*.
- `enable <unit>`
  `disable <unit>`
  `is-enabled <unit>`
  rend l'unité active ou inactive ou dit si elle l'est, au *boot* du système ;
- `halt`
  `poweroff`
  arrête le système.

  La différence entre *halt* et *poweroff* est que la seconde mettra le système hors tension après l'arrêt du système.

  halt et *poweroff* sont des appels aux cibles `halt.target` ou `poweroff.tarqet` ;
- `reboot` *reboote* le système ;

La suite dans la page de manuel `man systemctl`.

[16] De même, il existe des commandes pour lister les *sockets* (`list-sockets`) et les *timers* (`list-timers`) dont nous ne parlerons pas dans ces notes.

Rappel des bases réseau

Modèle **25**
Adresses **27**
Routage élémentaire **29**
Résolution d'adresse **30**
Noms **30**
Le coin des commandes **30**

Cette section se veut un rappel sur les concepts réseaux utiles au cours et à l'administrateurice [17] réseaux. Pour plus de détails, nous renvoyons le lecteur aux formations CISCO par exemple.

Modèle

En administration réseau nous nous contentons du modèle TCP-IP sous-modèle du modèle OSI.

- La couche application (*application layer*) concerne les applications utilisées sur le réseau : `http`, `smtp`, `ftp`, `telnet`…
- La couche transport (*host-to-host transport layer*) concerne le service de transport de données. Il y a deux protocoles :
  - **TCP** *transmission control protocol* service de transport de données avec détection et corrections d'erreurs, orienté connexion.

    TCP vérifie que le système distant est prêt à recevoir les données avant de les envoyer. Lorsque la poignée de main est faite, le système dit qu'il a établi la connexion.
  - **UDP** *user datagram protocol* service de transport de datagrammes sans connexion.

[17] Dans ces notes, l'écriture tente d'être inclusive. Un peu pour céder à la mode, un peu parce que ça « m'amuse » et un peu pour essayer d'inclure plus. J'essaierai de ne pas en abuser. Je troque généralement l'accord dit « du masculin l'emporte » contre l'accord de proximité et j'ajoute quelques nouveaux mots en évitant le point médian · qui fait peur ;-)

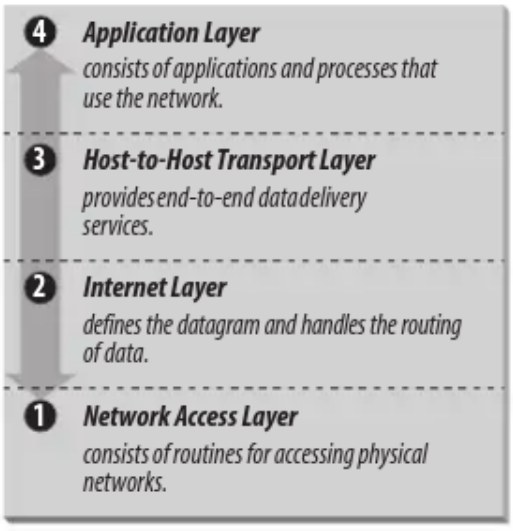


Figure : Modèle TCP-IP (source : TCP/IP Craig Hunt)

- La couche internet (*internet layer*) est la couche **IP**.

  IP est un protocole sans connexion. Ses rôles sont :

  – définir le datagramme (*datagram*) qui est l'unité de base pour la transmission sur internet;
  – définir le schema des adresses sur internet;
  – acheminer les données entre la couche *network* et la couche *transport*;
  – router les datagrammes;
  – s'occuper de la fragmentation et du réassemblage des datagrammes.

  ICMP *internet control message protocol* est un protocole inhérent à IP utilisant les datagrammes IP pour du contrôle, des rapports d'erreurs et de l'information.

  – *flow control* lorsque les datagrammes arrivent trop vite pour être traités, la destination ou une passerelle (*gateway*) intermédiaire envoie un *ICMP Source Quench Message* à l'expéditeur demandant d'interrompre l'envoi de manière temporaire;

  – *detecting unreachable destinations* lorsque le système détecte que la destination n'est pas atteignable, le système envoie un datagramme *Destination Unreachable Message*. Si la destination est un réseau ou un hôte, le message est envoyé par une passerelle intermédiaire. S'il s'agit d'un port inatteignable, c'est l'hôte qui envoie le message;

  – *redirecting routes* une passerelle envoie le message *ICMP Redirect Message* pour demander à un hôte d'utiliser une autre passerelle probablement parce que celle-ci est un meilleur choix. Ce message n'est envoyé que sur un même réseau.

  – *checking remote hosts* un hôte peut envoyer un message *ICMP Echo Message* pour tester si un système distant est *up* et opérationnel. Lorsqu'un système reçoit ce message il (peut) y répondre en renvoyant le paquet. `ping` utilise ce message.

- La couche réseau (*network access layer*) est la couche **Ethernet**.

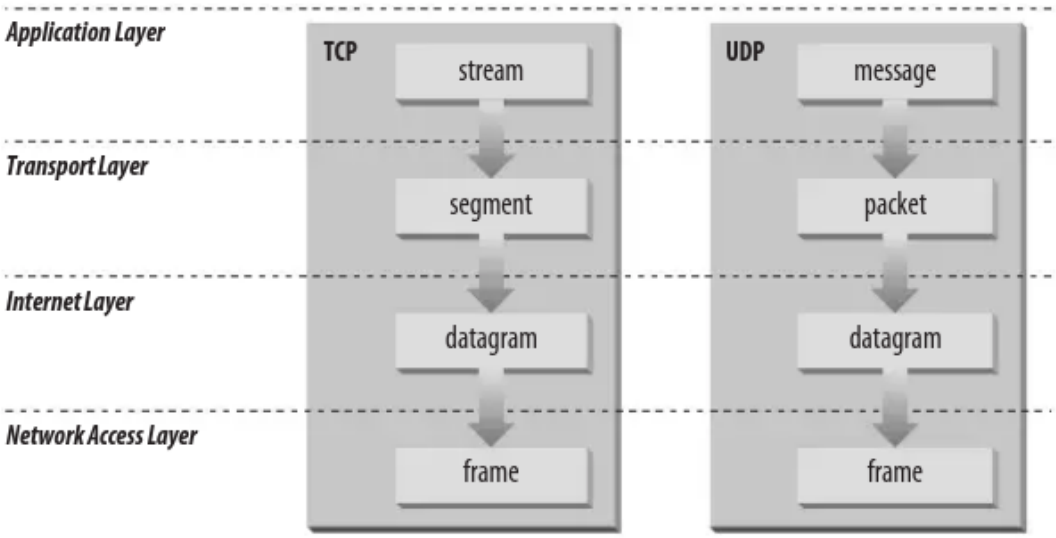


Figure : TCP et UDP dans le modèle TCP-Ip (source : TCP/IP Craig Hunt)

Adresses

Adresse IPv4

Une adresse IPv4 est une valeur de **32 bits** habituellement écrite comme 4 valeurs décimales séparées par un point « `.` » (*dotted decimal notation*) chaque nombre décimal représente 8 bits de l'adresse de 32 bits, chacun des nombres se trouvant dans l'intervalle 0-255.

Une adresse IPv4 se compose de deux parties :

- la première partie représente le numéro du **réseau** ;
- la seconde partie le numéro d'un **hôte** dans le réseau.

C'est le *masque* de réseau qui détermine **où** l'on coupe l'adresse en deux. Ce masque est composé d'un nombre de bits à 1 suivi d'un nombre de bits à 0.

Un réseau `/8` — anciennement, de classe A — a les 8 bits de gauche — les bits de poids forts — à 1. Son masque (*netmask*) est `11111111000000000000000000000000` en base 2 ou `255.0.0.0` en base 10 et en regroupant les bits par paquets de 8 (par byte).

Par exemple, l'IPv4 `10.0.0.1/8` représente :

- le réseau `10` ;
- la machine `0.0.1`.

Dans ce réseau numéro 10, il est possible d'avoir 16_777_214 hôtes (2^24-2). Les adresses `10.0.0.0` et `10.255.255.255` étant réservées respectivement pour le numéro de réseau et le *broadcast*.

L'adresse IPv4 de la boucle locale (*loopback*) est `127.0.0.1`.

Adresse IPv6

Une adresse IPv6 est une valeur de **128 bits** (16 bytes)[18]. La représentation hexadécimale regroupe les octets par 2 séparés par deux points « : ». Ce qui fait 8 groupes de 4 chiffres hexadécimaux. Par exemple :

```
2001:0db8:0000:85a3:0000:0000:\
  ac1f:8001
```

Il est permis de remplacer `0000` par `0` et de supprimer des groupes nuls tant que l'adresse ne devient pas ambiguë. L'adresse précédente peut s'écrire :

`2001:0db8:0:85a3::ac1f:8001`[19]

Les datagrammes IPv6 ont été simplifiés et ne comportent plus que 7 champs au lieu de 14 pour l'IPv4 ce qui accélère les traitements au niveau des routers.

Une adresse peut être un identifiant pour une interface (*unicast*) ou pour un ensemble d'interfaces (*multicast*). En ce sens, une interface peut avoir plusieurs adresses IPv6.

La notion de masque subsiste. Le préfixe est l'élément commun à toutes les adresses d'une même plage. La plupart du temps, le masque est `/64`.

Le *broadcast* est remplacé par le *multicast*.

Les types d'adresse IPv6 (*Internet Protocal Version 6 Address Space*, s.d.) (Goffinet, s.d.) :

- les adresses *unicast* désignant une destination unique (un hôte) :
  - la boucle locale (*localhost*) `::1/128` ;
  - l'adresse locale (*link local*) `FE80::/10` non routable utilisable au sein d'un réseau local ;
  - l'adresse publique (*global unicast*) `2000::/3` [20] ;
  - l'adresse privée (*unique local*) `FC00::/7` est l'équivalent des plages d'adresses privées en IPv4. Le 8e bit doit être positionné à 1 ce qui donne le préfique `FD00::/8` pour un réseau local ;
- les adresses *anycast* qui sont des adresses pour lesquels le chemin emprunté est au plus proche ou au plus efficient ;
- les adresses *multicast* `FF00::/8` remplacent les adresses de *broadcast* désignant potentiellement plusieurs destinations :

[18] Ce qui fait 2^{128} adresses possibles, soit $\pm 3\,10^{38}$. Pour l'image, si la surface de la terre était recouverte d'ordinateurs ayant chacun, une IPv6, il serait possible d'allouer $7\,10^{23}$ adresses IP / m^2 (*source commentcamarche.net*).

[19] Les 5^e et 6^e octets sont à 0 et peuvent être remplacés par `:0:` et les octets de 9 a 12 peuvent être remplacés par `::`.

[20] C'est bien `/3` et donc `200`. Les adresses unicast globales `2001::/16` peuvent être réservées et sont allouées par bloc `/23` à `/12` par l'IANA(*iana (internet assigned numbers authority), the global coordination of the DNS Root, IP addressing…*, s.d.). L'IANA alloue aux RIR (*regional internet registry*) — RIPE NCC pour l'europe — qui allouent à leur tour au LIR (*local internet registry*) qui sont généralement également fournisseur d'accès. Les adresses `2002::/16` sont utilisées par *6to4* pour acheminer du trafic IPv6 vers IPv6 à travers IPv4. Les autres adresses sont réservées à un usage ultérieure. Ceci explique pourquoi les adresses unicast globales IPv6 sont de la forme `2001…`.

  - les 4 bits les moins significatifs du 2e byte (`FF0s::`) identifient la portée de l'adresse :

    `1` locale à l'hôte
    `2` locale au lien
    `5` locale au site
    `8` locale à l'organisation
    `e` globale

Il existe plusieurs techniques pour assigner une adresse en fixant l'identifiant d'interface (les 64 bits de poids faibles) :

- l'identifiant d'interface peut être fixé de manière arbitraire ;
- l'identifiant d'interface peut être configuré automatiquement :
  - déduit de l'adresse MAC (cfr. RFC 4862) en utilisant par exemple MAC EUI-64 (cfr. RFC 4291). Comme ces techniques exposent l'adresse MAC, elles sont déconseillées par IETF depuis 2017 ;
  - autoconfiguration basée sur une clé secrète et le préfixe réseau (cfr. RFC 7217)
  - autoconfiguration par un tirage pseudo-aléatoire (cfr. RFC 4941).
  - obtenu **dynamiquement** à partir de `DHCPv6`

ARP est remplacé par **ND** (*neighbor discovery*) et des messages `ICMPv6`.

L'adresse IPv6 de la boucle locale (*loopback*) est `::1`.

Routage élémentaire

Les passerelles routent les données à travers le réseau. Pour ce faire, les composants réseaux, les passerelles et les hôtes, doivent prendre des décisions de routage.

Pour la plupart des hôtes les décisions sont simples:

- si la destination est sur le réseau local, les données sont délivrées à l'hôte;
- si la destination est sur un réseau distant, les données sont transmises à la passerelle locale.

Une telle table de routage peut avoir cette allure en IPv4:

```
alice@harmony:~$ ip route
default via 192.168.1.1 \
    dev eth0 onlink
192.168.1.0/24 dev eth0 proto kernel \
    scope link src 192.168.1.11 \
    metric 100
```

Remarque: depuis le noyau 3.6, il n'y a plus de cache pour le routage IPv4.

Résolution d'adresse

L'adresse IP et la table de routage adresse un datagramme à un réseau cependant le datagramme doit passer par la couche physique. L'adresse IP doit être mappée à une adresse physique dépendante du réseau physique ; habituellement **ethernet**.

Le protocole faisant la traduction IP → Ethernet est **ARP** *address resolution protocol* pour IPv4 et **NDP** *neighbor discovery protocol* pour IPv6.

ARP maintient une table de correspondance entre adresse IP et adresse Ethernet ou adresse MAC. Lorsque ARP reçoit une demande de traduction d'une adresse IP, il regarde dans sa table. Si l'adresse s'y trouve, il la retourne sinon, ARP broadcaste un packet à tous les hôtes sur le segment. Le paquet contient l'IP concernée. Si un hôte a cette IP, il répond et la correspondance IP-Ethernet est cachée — au sens mise en cache — dans la table ARP.

Noms

Chaque système porte un nom, son nom d'hôte (*hostname*) qui associé au nom de domaine (*domainname*) devrait être unique.

Sur une machine linux, le nom d'hôte est renseigné dans `/etc/hostname`. Il peut être lu ou changé *via* la commande `hostname`.

Le nom de domaine est donné par la commande `dnsdomainname` pour obtenir le nom dns et `[yp]domainname` pour obtenir le nom de domaine au sens NIS/YP (*yellow pages*)[21].

La méthode recommandée pour positionner le **fqdn** (*fully qualified domain name*) est d'écrire un *alias* au nom d'hôte dans le fichier `/etc/hosts`. Par exemple :

```
127.0.0.1 harmony.example.org harmony
```

Par défaut le fichier `/etc/hosts` est lu avant de faire une requête DNS.

Le coin des commandes

`ip`

`ip` est la commande à tout faire pour la configuration du réseau[22].

Elle se présente comme `ip <object> [<command>]` par défaut, c'est la commande `show` (*alias* pour `list`) qui est exécutée. Les objets peuvent être abrégés.

[21] NIS (*network information system*) et ses *yellow pages* ne sera pas abordé dans ces notes.

[22] Cette commande remplace `ifconfig` avec `ip address`, `route` avec `ip route` et `arp` avec `ip neighbour`.

`ip address` (ou `ip a`) pour l'adressage IPv4 ou IPv6.

```
ip a
ip -6 a
ip a show dev eth0
ip a delete 2001:0db8:85a3…/64
  dev eth0
```

- montre toutes les interfaces réseaux (IPv4 et IPv6)
- montre toutes les interfaces réseaux en se limitant à IPv6
- montre l'interface *eth*
- efface l'adresse précisée de l'interface eth0

`ip route` (ou `ip r`) pour manipuler la table de routage.

```
ip r
ip r add default via 10.0.0.1
  dev eth0
ip -6 r add 2001:db8:1::/64 \
    via 2001:db8:42::1 dev eth0
```

- montre la table de routage
- ajoute une route par défaut via `10.0.0.1` sur l'interface *eth0*
- ajoute une route pour le réseau `2001:db8:1::/64` via `2001:db8:42::1` sur l'interface *eth0*

`ip neighbour` (ou `ip neigh` ou `ip n`) pour manipuler les tables des voisins. Pour IPv4, il s'agit de la table ARP.

```
ip n
ip n flush dev eth0
```

- montre la table
- supprime les entrées de la table pour *eth0*

`hostname`

`hostname` montre ou positionne le nom d'hôte de la machine.

```
hostname
hostname --fqdn
```

- montre le nom d'hôte
- montre le nom d'hôte ainsi que le domaine

`netstat`

`netstat` affiche les connexions réseaux, les tables de routage, les statistiques des interfaces…

Voici quelques options[23] :

- `-r`, `--route` permet l'affichage des tables de routages
- `-i`, `--interfaces` montre une table de toutes les interfaces réseau
- `-6` par défaut les commandes sont en IPv6, cette option demande l'IPv6
- `-p`, `--programs` affiche le nom et le *pid* des processus propriétaire de chaque socket
- `-a`, `--all` affiche tous les sockets (par défaut connectés)
- `-e`, `--extend` donne quelques infos supplémentaires
- `-c`, `--continuous` pour afficher de manière continue
- `-l`, `--listening` affiche les sockets en écoute

```
netstat -rn
netstat -r6
netstat -p
netstat -lapute
```

- affiche les tables de routage IPv4 au format numérique

[23] Exceptionnellement, je conseille de préférer l'aide en ligne (`netstat --help`) à la page de manuel.

- affiche les tables de routage IPv6
- affiche tous les sockets ouverts
- affiche un peu tout[24]

`dig`

`dig [@server] [name] [type) [class]` est une commande pour l'interrogation d'un serveur DNS.

- `-4`, `-6` demande de faire la requête en IPv4 ou en IPv6
- `<type>` représente le type de requête demandé : A, AAAA, NS…
- `<name>` est le nom de la resource : `example.org`…
- `@server` précise le serveur DNS utilisé. Si aucun serveur n'est renseigné, c'est le serveur défini dans `/etc/resolv.conf` que est utilisé
- `+trace` fait la demande à partir du serveur de nom racine et montre les réponses de chaque serveur de nom.

```
dig
dig example.org
dig example.org MX
dig @127.0.0.1 +trace example.org
```

- montre les serveurs racines
- montre l'adresse IP associée au nom `example.org`
- montre le champ `MX` associé au nom `example.org`
- demande au serveur local l'adresse IP associée à `example.org` avec une trace des différentes requêtes.

[24] `netstat` est sans doute la commande pour laquelle moult moyens mnémotechniques existent dans lequel chaque personne peut faire son choix: `-alpe --ip`, `-taupe`, `-lapute`.

Le shell

Commandes `shell` **34**
Les expansions **35**
Les redirections **37**
Les tests **38**
L'historique **39**
Les commandes internes de `bash` **40**
Script `shell` **41**
Astuces et raccourcis `bash` **42**
Le coin des commandes **43**
head 43
tail 43
grep 43

Le *shell* est l'interprèteur de commandes. C'est le programme faisant l'interface entre l'utilisateurice et la machine. C'est une interface en ligne de commandes accessible via une console et permettant à l'utilisateurice de lancer des commandes en entrant un texte.

Le premier *shell* `sh` est dû à Thompson Shell (1971) et a été remplacé par le shell de Stephen Bourne (1977). Brian Fox réécrit ce shell et l'appelle *borne again shell* `bash` en 1988. C'est le shell le plus répandu bien qu'il en existe d'autres: `csh`, `tcsh`, `ksh`, `zsh`…

Dans la suite, nous nous intéressons à `bash`.

`bash` peut être invoqué de différentes manières :

- comme shell de login (*login shell*), si le premier caractère de son argument 0 est un tiret « - » ou s'il est invoqué avec l'option `--login`;
- comme shell interactif (*interactive shell*) si l'entrée et la sortie standards sont connectées à un shell ou s'il est invoqué avec l'option `-i`[25]

Lorsque `bash` est lancé, il lit des fichiers d'initialisation :

- pour un shell de login (interactif ou non), le fichier `/etc/profile` — s'il existe — est exécuté ;
- pour un shell de login (interactif ou non), `bash` cherche dans l'ordre les fichiers `~/.bash_profile`, `~/.bash_login` et `~/.profile` et exécute le premier qu'il trouve (l'option `--noprofile` empêche ce comportement);
- pour un shell interactif (mais pas de login), `bash` exécute les fichiers `/etc/bash.bashrc` et `~/.bashrc` s'ils

[25]La variable `$-` peut être consultée pour voir les options passées au shell courant.

existent (l'option `--norc` empêche ce comportement);

Lors de l'*exit*, le fichier `~/.bash_logout` est exécuté s'il existe.

Commandes `shell`

Une **commande simple** est une séquence optionnelle d'assignation de variables suivie de mots et de redirections éventuelles. La commande se termine par un opérateur de contrôle par exemple : `||`, `&&`, `&`, `;`, `|`, `|&`, `<new line>`… Le premier mot est la commande a exécuter et les mots suivants sont passés comme arguments à la commande.

Un *pipe* est une séquence de commandes séparées par le caractère *pipe* « `|` ».

```
command1 | command1
```

la sortie standard de la première commande `command1` est connectée à l'entrée standard de la deuxième commande `command2`.

Si les caractères « `|&` » sont utilisés, la sortie standard **et** la sortie d'erreur standard entre dans le *pipe*.

Un commande peut se terminer par « `&` ». Dans ce cas la commande est exécutée en tâche de fond (*background*) et le shell rend la main.

Les commandes peuvent être séparées par un ET « `&&` » ou un OU « `||` ».

```
command1 && command2
```

- `command2` sera exécutée si et seulement si `command1` retourne *true*.

```
command1 || command2
```

- `command2` sera exécutée si `command1` retourne *false*.

Une **commande composée** est l'une des constructions suivantes (la liste n'est pas complète, cfr. `man bash`) :

- `(list)` une liste de commandes entre parenthèses est exécutée dans un sous-shell ;
- `{ list; }` une liste de commandes exécutées dans le shell courant. Peut être vue comme un bloc ;
- `[[ expression ]]` évalue l'expression conditionnelle entre double crochets « `[[ ]]` » ;

  ```
  $ [[ -a /tmp/file ]] \
    && echo "File exist"
  ```

  – affiche "File exist"… si le fichier existe.

- `for <name> in word… ; do <list>; done`

  ```
  $ for i in $(seq 10); do
    echo $i; done
  $ for i in one two three; do
      touch $i;
    done
  ```

  – affiche les valeurs de 1 à 10
  – crée les trois fichiers `one`, `two` et `three`

- `for (( expr1 ; expr2 ; expr3 )); do <list>; done`

```
$ for ((i=0 ; i < 10 ; i++)) ; do
>   echo $i;
> done
```

– affiche les valeurs de 0 à 9

- `if <list>; then <list>; [else <list>;] fi`

```
$ if [ $DISPLAY = ":0" ] ; then
>   echo "X is launch" ;
> fi
```

– affiche “X is launch” si la variable `DISPLAY` vaut “:0”

`bash` permet de déclarer des **variables**.

```
name=value
```

Notons qu’il ne faut pas d’espace de part et d’autre du signe « = ».

Les **paramètres positionnels** sont les arguments du shell lors de son invocation : `$1`, `$2`…

Les **paramètres spéciaux** sont repris dans le tableau suivant :

Param	Description
`$*`	Tous les paramètres positionnels. Entre guillemets, représente un seul mot : `$1_$2_$3`… (où _ représente un espace)
`$@`	Tous les paramètres positionnels. Entre guillemets, représente plusieurs mots : `$1 $2 $3`…
`$#`	Nombre de paramètres (décimal)
`$?`	Code de retour de la dernière commande.
`$-`	Liste des options avec lesquelles le shell a été invoqué.
`$$`	*pid* du shell.
`$!`	*pid* de la dernière commande exécutée en arrière plan.
`$0`	Nom du script (ou du shell).

Les expansions

Toute une série d’**expansions** se font dans **cet ordre**:

1. expansion des accolades (*brace expansion*),
2. développement du tilde « ~ » (*tilde expansion*),
3. remplacement des paramètres et variables (*parameter and variable expansion*),
4. substitution de commandes (*command substitution*),
5. évaluation arithmétique (*arithmetic expansion*),
6. découpage des mots (*word splitting*) et
7. développement des noms de fichiers (*pathname expansion*).

1 L’**expansion des accolades** permet

la création de chaines quelconques sous la forme d'un préambule facultatif, suivi de chaines entre accolades et séparées par des virgules (sans espace blanc), le tout éventuellement suivi d'un postambule.

```
$ echo a{b,c,d}e
abe ace ade
```

Pratique par exemple pour créer plusieurs répertoires
`mkdir /tmp/dir-{one,two}`.

2 Le **développement du tilde** de manière simplifiée se remplace par la valeur de `HOME`. Tous les caractères précédent de premier slash « `/` » sont considérés comme un *login*.

```
$ echo ~
/home/alice
$ echo ~bob
/home/bob
```

3 Le **remplacement des paramètres** s'associe au symbole dollar « `$` » et aux accolades « `{}` ». Même si une variable peut s'écrire `$var` il est conseillé de l'écrire `${var}`

`${parametre}` est remplacé par la valeur du paramètre.

Le remplacement des paramètres est plus large que cette simple « valeur de variable » et les principaux remplacement sont repris dans le tableau suivant (cfr. *man pages* pour la totalité) :

Remplacement de paramètres	**Description**
`${parametre:-mot}`	Donne la valeur du paramètre. Si le paramètre n'est pas défini, donne la valeur de *mot*
`${parametre:=mot}`	Donne la valeur du paramètre. Si le paramètre n'est pas défini, donne la valeur de *mot* et initialise le paramètre avec la valeur de *mot*
`${parametre:?mot}`	Donne la valeur du paramètre s'il existe sinon affiche *mot* comme message d'erreur.
`${parametre:+mot}`	Donne la valeur du *mot* si le paramètre existe. Sinon ne retourne rien.
`${#parametre}`	Donne la longueur de la valeur du paramètre.
`${parametre:offset}`	Donne la valeur du paramètre jusqu'à *length* en
`${parametre:offset:length}`	commençant au caractère d'indice *offset*.
`${!prefix*}` `${!prefix@}`	Donne toutes les variables commençant par *prefix*.
`${parametre#mot}`	Supprime le schema correspondant au début.
`${parametre##mot}`	
`${parametre%mot}`	Supprime le schema correspondant à la fin.
`${parametre%%mot}`	
`${parametre/pattern/string}`	Substitue *pattern* par *string*.

4 La **substitution de commandes** s'associe au symbole dollar « `$` » et aux parenthèses « `()` »[26].

`$(command)` est remplacé par le retour de la commande.

La commande est exécutée dans un sous-shell.

5 L'**évaluation arithmétique** s'associe au symbole dollar « `$` » et à la double paires de parenthèses « `(())` ».

`$((expression))` est remplacé par la valeur de l'expression arithmétique[27].

6 Le **découpage des mots** se fait à la suite des expansions précédentes et tous les mots sont séparés en fonction de l'espace blanc (sauf si `IFS` a été modifié). Pour que des valeurs ne soient pas séparées, il suffit de les entourer de guillemets « `"` ».

7 Le **développement des noms de fichiers** consiste à rechercher dans les mots les *jokers* éventuels : `*`, `?` et `[]`.

- `*` correspond à n'importe quelle chaine;
- `?` correspond à n'importe quel caractère. Un seul;
- `[…]` correspond à une suite de caractères, un intervalle ou à une classe de caractères.

Les redirections

Par défaut l'entrée d'une commande, est l'entrée standard — le clavier — et les sorties de la commande sont la sortie standard — le terminal. Ce comportement peut être modifié en **redirigeant** ces canaux[28].

`<mot` lit le fichier (ouvert en lecture) *mot* comme entrée standard.

`>mot` redirige la sortie standard dans le fichier *mot* (ouvert en écriture).

`2>mot` redirige la sortie d'erreur standard dans le fichier *mot* (ouvert en écriture).

L'utilisation de « `>>` » redirige une sortie en **ajout** dans le fichier.

Il est possible de rediriger deux sorties dans un même fichier. Dans ce cas, la première redirection redirige dans le fichier (`>mot`) et la seconde en faisant référence à la redirection précédente avec l'esperluette (`2>&1`).

Pour rediriger la sortie standard et la sortie d'erreur standard dans le même fichier :

```
$ ls > out 2>&1
```

[26] La substitution de commande se fait également en utilisant les guillemets inverses (*back quotes*) « ` ». Il est conseillé de ne plus utiliser cette notation même si elle est encore fréquente dans la littérature.

[27] Le format `$[expression]` est déprécié et va disparaitre dans les prochaines versions de `bash`. À oublier donc…

[28] Ces notes présentent la redirection des descripteurs de fichiers (*file descriptor*) 0, 1 et 2, l'entrée standard, la sortie standard et la sortie d'erreur standard. Il est possible de rediriger d'autres descripteurs de fichier. Voir `man bash`.

Les tests

Les **expressions conditionnelles** sont utilisées pour tester les fichiers, les chaines de caractères et l'arithmétique. Ces conditions sont testées avec la commande « `[[ ]]` » ou « `[]` » ou encore la commande `test`.

`bash` peut évaluer des **expressions arithmétiques** par exemple lors de l'expansion arithmétique. Les opérateurs habituels —y compris ceux de post|pré-inc|décrémentation ainsi que l'opérateur conditionnel— sont disponibles.

Par exemple :

```
$ for (( i=0 ; i < 10 ; i++ )) ; do
>   echo $i;
> done
```

`bash` peut faire des tests sur les fichiers. Par exemple :

- `-a file` teste l'existence du fichier;
- `-d file` teste l'existence et si c'est un répertoire;
- `-f file` teste l'existence et si c'est un fichier régulier;
- `-r file` teste l'existence et si le fichier est *readable*;
- `-w file` teste l'existence et si le fichier est *writable*;
- `-x file` teste l'existence et si le fichier est *excutable*;
- `file1 -nt file1` teste si *file1* est plus récent (*newer than*) *file2*;

`bash` peut faire des tests sur les chaines de caractères. En voici quelques uns :

- `-n string` vrai si la longueur de la chaine est non nulle;
- `-z string` vrai si la longueur de la chaine est nulle;
- `string1 = string2` vrai si les deux chaines sont égales (`==` fonctionne aussi);
- `string1 != string2` vrai si les deux chaines sont différentes;
- `string1 > string2` vrai si les deux chaines sont triées lexicographiquement;

Exemples :

```
$ if [ -d /tmp ] ; then
>   echo "dir exist";
> fi
```

```
$ if test -d /tmp ; then
>   echo "dir exist";
> fi
```

```
$ VAR=file.txt
$ if test -z ${VAR} ; then
>   echo "foo";
> else
>   echo "bar";
> fi
```

L'historique

`bash` possède un historique des dernières commandes entrées. Par défaut cet historique mémorise 500 commandes. `echo ${HISTSIZE}` conserve cette valeur.

Les flèches haut et bas permettent de naviguer dans cette historique.

Au delà de cette utilisation basique, voici quelques usages de l'historique.

- `history` affiche l'historique. Associée à un `grep`, il est facile de retrouver une commande;
- `!n` réexécute la ligne `n`;
- `!-n` réexécute la ligne se référent à la commande courante - `n`;
- `!!` réexécute la dernière commande;
- `!string` réexécute la commande la plus récente commençant par la chaine `string`;
- `^string1^string2` répète la commande précédente en remplaçant `string1` par `string2`;

Il est également possible d'utiliser les évènements précédents autrement que de simplement les exécuter. Là où `!!` execute la dernière commande, `!!:$` représente le dernier mot de la dernière commande par exemple.

Derrière ce `:` peuvent se trouver d'autres « désigneurs » de mots (*word designators*). En voici quelques uns :

- `n` le `n`^{e} mot de la commande ;
- `^` le premier ;
- `$` le dernier ;
- `i-j` du `i`^{e} au `j`^{e} mot ;
- `*` tous les mots sauf la commande elle-même

Par exemple `!!:$`. Cette notation peut être raccourcie : `!!:$` par `!$` et `!!:^` par `!^`, etc.

Encore derrière ces *word designators* peuvent se trouver des modificateurs (*modifiers*) également précédés de deux points « `:` ». En voici quelques uns :

- `r` retire le suffixe et laisse le nom de base (*basename*) ;
- `p` affiche la commande mais ne l'exécute pas ;
- `s/old/new` remplace la première occurrence de `old` par `new` ;
- `gs/old/new` remplace toutes les occurrences de `old` par `new` ;

Voici quelques usages dont certains sont issus de *The geek stuff*[29] :

```
$ ls /etc/apache2/apache2.conf
$ vim !!:$
```

```
$ xpdf /elsewhere/book.pdf
$ vim !!:s/pdf/md
```

```
$ cp pam.conf pam.bak
$ vi !^
```

```
$ cp ~/longname.conf \
  /really/a/very/long/path/other.conf
$ chmod go-rw !!:2
chmod go-rw \
  /really/a/very/long/path/other.conf
$ ls -l !cp:2
ls -l /really/a/very/long/path/\
  other.conf
-rw------- 1 bob bob 0 fév 24 10:22 \
  /really/a/very/long/path/other.conf
```

[29] https://www.thegeekstuff.com/2008/08/15-examples-to-master-linux-command-line-history

Les commandes internes de `bash`

La plupart des commandes linux sont des binaires ou des scripts répartis dans différents répertoires du *filesystem*. Il existe toutefois une série de commandes qui sont des **commandes internes** à `bash`. Ces commandes sont directement interprètées pas le shell sans lancer un processus spécifique exécutant la commande.

En voici quelques unes, pour la liste complète, consulter le manuel `man bash` :

- `source filename` lit et exécute les commandes du fichier *filename* ;

```
$ source ~/.bashrc
```

- `alias [name=value]`
  - sans argument affiche la liste des *alias* définis;
  - lorsqu'un argument est fourni, défini l'*alias*

```
alias ll="ls -l"
```

- `bg jobid` place le *job jobid* (liste accessible par `jobs`) en tâche de fond (*background*) comme s'il avait été lancé avec `&` ;

```
$ mycommand
[Ctrl-z]
$ bg 1
```

- `cd [<dir>]` change de répertoire (*change directory*) vers le répertoire *dir* s'il existe, sinon, vers `HOME` ;

  `cd -` se rend dans le répertoire dans lequel l'*user* se trouvait précédemment. Pratique pour faire un aller-retour.

- `echo [-neE] [arg]` affiche *arg*. L'option `-n` supprime la passage à la ligne de fin, `-e` active l'interprètation des séquences d'échappement et `-E` la désactive. Cette option est l'option par défaut ;

- `exit [n]` quitte le shell avec la valeur de retour *n* si elle est fournée, 0 sinon ;

- `fg jobid` place le *job* en avant plan (*foreground*) ;

- `help [-dms] [pattern]` donne de l'aide sur une commande interne. L'option `-d` donne une description courte, `-m` donne l'aide dans un format *man page* et `-s` donne un usage court ;

- `history` affiche l'historique complet,
  `history -c` efface tout l'historique,
  `history -d offset` efface l'entrée en position *offset*,
  `history -d start-end` efface les entrées de *start* à *end*,

- `kill [-s sigspec | -n signum | -sigspec] [pid | jobspec]` envoie le signal *sigspec* ou *signum* au processus *pid* ou *jobspec*,
  `kill -l | -L` liste tous les signaux

- `pwd` affiche le réportoire courant ;

Script `shell`

Un script shell n'est qu'une suite de commandes shell placées dans un fichier.

Un script peut porter l'extension `.sh` ou `.bash` mais elle n'est pas obligatoire.

Un script shell commence par le nom du shell qui doit être utilisé. Cette première ligne, à l'allure suivante, s'appelle le ***shebang*** :

```
#! /bin/bash
```

C'est une bonne pratique de faire immédiatement suivre ce *shebang* d'un commentaire précisant la fonction du script, l'auteur… et puis une fonction en précisant l'usage. Par exemple :

```
usage() {
  echo -e "$(basename ${0}) options…"
  echo -e "…"
}
```

Une fonction se définit comme précédemment et s'appelle simplement en donnant son nom (sans les parenthèse).

```
usage
```

Un script shell se trouvant par exemple dans le fichier `yascript.sh` peut s'exécuter par :

- `bash yascript.sh`[30] ou;
- `./yascript.sh` si le fichier a été rendu exécutable au préalable.

Quelques options sont pertinentes pour écrire un script plus sécurisé.

`set -e` entraine que le script quitte après une commande en erreur, sans continuer.

```
badcommand
echo "end"
```

- comme la commande n'existe pas, *end* ne sera pas affiché.

`set -o pipefail` entraine que le code de retour d'un pipe ne soit pas celui de la dernière commande mais celui de la dernière commande en erreur.

Associée à l'option `-e`, permet au script de quitter dès qu'une erreur survient… même dans un *pipe*. Par exemple :

```
$ bash -c  "badcommand1
> | echo foo;
> badcommand2;
> echo bar"
foo
bash: badcommand1 : commande introu…
bash: badcommand2 : commande introu…
bar
```

```
$ bash -ceo pipefail  "badcommand1
> | echo foo;
> badcommand2;
> echo bar"
foo
bash: badcommand1 : commande introu…
```

`set -u` entraine que le script quitte s'il rencontre une variable qui n'existe pas.

Le nom du script et ses paramètres sont accessibles par `$0` pour le nom du script puis, `$1`, `$2`… pour les paramètres.

L'accès à une variable devrait se faire entre accolades « `{}` » **et** entre guillemets « `""` ».

Un script devrait | pourrait avoir l'allure suivante :

[30] Lancé avec l'option `-x`, `bash` exécutera le script en mode *debug*.

```bash
#!/bin/bash
#
# Demo script.
#
set -eo pipefail

readonly VAR=${1}

set -u

usage() {
  echo -e "$(basename ${0}) options…"
  echo -e "…"
}

# param verification
if [ -z "${VAR}" ]; then
  echo "error message…"
  usage
  exit 1
fi

# now, do effective stuff
```

Astuces et raccourcis `bash`

- `[Esc] t` intervertit les deux derniers paramètres d'une commande.

  Pratique lors de l'écriture de `systemctl apache2 start` au lieu de `systemctl start apache2`.

- Le paquet `bash-completion` propose une autocomplétion intelligente pour toutes les commandes. À installer.

- `[Ctrl-r]<pattern>` affiche la première commande correspondant au schéma (*pattern*) disponible dans l'historique.

- `[Esc] .` écrit la dernière commande (sans le réexécuter immédiatement).

Le coin des commandes

head

`head [-n]` affiche les *n* premières lignes d'un fichier. Par défaut, *n* vaut 10.

tail

`tail [-n]` affiche les *n* dernières lignes d'un fichier. Par défaut, *n* vaut 10. D'autres options sont disponibles (cfr. `man tail`)

- l'option `-f` affiche les dernières lignes en continu, pratique pour des fichiers de logs ;

```
# tail -f /val/log/syslog
```

grep

`grep [options] patterns [file…]` recherche chaque *patterns* dans chaque fichier *file*. `grep` peut lire sur l'entrée standard `-`. Voici quelques options (pour d'autres options, `man grep`) :

- `-i` *ignore case* ;
- `-v` inverse la sélection, donne les lignes qui ne correspondent pas au *pattern* ;
- `-c` n'affiche pas les lignes mais les comptes (*count*) ;
- `-m num` arrête de chercher après *num* lignes correspondant au *pattern* ;
- `-r` demande une recherche récursive à partir du répertoire renseigné, dans tous les sous-répertoires…

SSH - Secure shell

Connexion SSH par échange de clés 45
Éléments de sécurité d'un serveur SSH 45
ports knocking 46
Le coin des commandes 47
openssl 47
ssh-keygen 47
ssh-copy-id 47
ssh-agent 47
ssh-add 48
ssh 48
knock 48

Secure Shell (SSH) est un protocole de communication sécurisé qui permet aux utilisateurs et aux utilisatrices de se connecter à un serveur distant et d'exécuter des commandes à distances. SSH permet d'obtenir un terminal à distance de manière sécurisée.

L'ancêtre non sécurisé de `ssh` est `rsh` (*remote shell*).

SSH garantit à la fois l'**authenticité** du serveur et la **chiffrement de la connection**.

Le serveur possède une paire de clés cryptographiques ce qui (peut) garantit que l'on se connecte au serveur légitime. Lors de la première connexion, le serveur présente sa clé publique et le client peut l'accepter et la stocker (dans sons fichier `~/.ssh/known_hosts`). Pour toutes les autres connexions, le client vérifiera que la clé présentée pour ce serveur est bien celle qu'il détient.

Une fois la clé présentée, le client challenge le serveur en lui présentant un message chiffré avec la clé publique du serveur que celui-ci doit pouvoir déchiffrer s'il possède bien correspondante. À ce stade la tentative de connexion se fait bien auprès du bon serveur. Les algorithmes utilisés sont des algorithmes de cryptographie **asymétrique** tels que `DSA` (obsolète), `RSA` de taille 4096, `ed25519` ou encore, `ecdsa`.

Remarque : lors de la première connexion, il est possible de vérifier que l'empreinte fournie par le serveur est bien *la bonne* simplement en la diffusant sur un autre canal que la connexion `ssh`.

Dès que l'authenticité du serveur est montrée, le client et le serveur doivent s'échanger une clé de session sur un canal non sûr. Ils utilisent pour ça un algorithme de cryptographie **d'échange de clés** utilisant un des protocoles *Diffie-Hellman* ou *Elliptic Curve Diffie-Hellman*.

Une fois la clé de session échangée, le

client et le serveur utilisent un algorithme de cryptographie **symétrique** pour authentifier le client et communiquer en chiffrant les messages. Les algorithmes utilisés garantissent la **confidentialité** des messages et leur **intégrité**. L'intégrité d'un message est la garantie qu'il n'a pas été altéré en chemin. Pour ce faire, on associe au message un *code d'authentification du message* (MAC, *message authentication code*). C'est une sorte de *hash* du message qui circule avec lui. Les algorithmes de cryptographie symétrique sont par exemple `AES`, `blowfish`, `chacha20`… et `AES`, par exemple, embarque avec lui la signature du message.

Connexion SSH par échange de clés

Plutôt que de renseigner son mot de passe lors de chaque connexion, SSH peut authentifier le client grâce à une paire de clés. Dans ce cas, le client dépose au préalable sa clé publique sur le serveur et lors de la connexion, le serveur challenge le client afin d'avoir confirmation qu'il possède bien la clé privée.

Pour ce faire, il est nécessaire de :

- générer une paire de clés sur le client à l'aide de `ssh-keygen`;

```
ssh-keygen -t rsa -b 4096
  -C "user@example.org"
```

- déposer la clé publique sur le serveur (nécessite une authentification par mot de passe[31]);

```
ssh-copy-id -i ~/.ssh/id_rsa.pub
  user@host
```

- configurer le serveur pour qu'il accepte l'authentification par échange de clés en modifiant la configuration du serveur:
  - `PubkeyAuthentication yes` autorise l'authentification par échange de clés
  - `PasswordAuthentication no` interdit l'authentication par mot de passe.

Éléments de sécurité d'un serveur SSH

Voici quelques paramètres de configuration nécessaires pour tout serveur SSH.

`PermetRootLogin no` n'autorise pas l'utilisateur *root* à se loguer. C'est une bonne pratique de se connecter en tant qu'utilisateur normal et devenir *root* —

[31] Il est possible de donner cette clé par un autre canal et de la copier manuellement sur le serveur dans le fichier `~user/.ssh/authorized_keys` où `~user` est le répertoire *home* du l'utilisateurice qui tente la connexion.

ou utiliser `sudo` — si nécessaire.

Cette option peut être combinée à `AllowsUsers <users>` qui liste les comptes autorisés à se connecter au serveur.

`ClientAliveInterval 300` fixe un *idle timeout* c'est-à-dire un temps après lequel le serveur va envoyer un message au client demandant une réponse. Après `n` requêtes — où `n` est précisé dans `ClientAliveCountMax n` — le client est déconnecté.

Pour éviter l'utilisation des mots de passe, il est conseillé de privilégier la connexion par échange de clés à celle par mot de passe.

`Port 22`

Une question se pose souvent au sujet du port utilisé pour le service SSH. Le port par défaut est le port **22**. Faut-il utiliser ce port et essuyer moults tentatives de connexions faites par divers attaquants ou changer de port ? **Oui et non**.

Changer le port soulage les logs et évite les nombreuses tentatives de connexions qui pourraient être prise en charge par un IDS (*intrusion detection system*). Changer le port ne préserve pas d'un scan complet sur la machine qui va trouver sur quel port le service SSH tourne.

ports knocking

Le *ports knocking* est une technique utilisée pour sécuriser un serveur (SSH ou autre). Une connexion est tentée à une séquence de ports pour ouvrir ou fermer un port d'application ou de service. Dès lors qu'une IP tente une connexion sur chacun des ports de la séquence, le serveur ouvre le port de l'application (par exemple le port 22 pour SSH) pendant une durée limitée *pour cette IP particulière* qui peut alors se connecter au service.

Le service `knockd` permet d'automatiser facilement cette technique.

Voici un exemple de configuration (sur base des pages de manuel) :

```
[options]
    logfile = /var/log/knockd.log

[ssh]
    sequence      = 2222,3333,4444
    seq_timeout   = 5
    start_command = ufw allow from %IP% to any port 22
    tcpflags      = syn
    cmd_timeout   = 10
    stop_command  = ufw delete allow from %IP% to any port 22
```

Une tentative de connexion sur chacun des ports 2222, 3333, 4444 (dans cet ordre) entraine l'exécution de la commande *start_command* qui ouvre le port pour l'IP en question. La règle est ensuite supprimée.

Une fois configuré, la tentative de con-

nexion en SSH doit au préalable frapper à la porte *toc toc toc* :

```
knock -v host.example.org
   2222 3333 4444
ssh user@host.example.org
```

Le coin des commandes

openssl

`openssl` est un peu la commande à tout faire en terme de gestion des clés cryptographiques.

`openssl genrsa -out key.pem 2048` génère une clé RSA de taille 2048.

`openssl rsa in key.pem -pubout -out key_pub.pem` extrait la clé publique de la clé privée.

`echo "Message" | openssl pkeyutl -encrypt -inkey key_pub.pem -pubin -out message.enc` chiffre le message avec la clé publique.

`openssl pkeyutl -decrypt -inkey key.pem -pubin -in message.enc` déchiffre le message avec la clé privée.

`echo "Message" | openssl enc -e -aes-256-cbc -out message.enc` chiffre le message avec un algorithme de chiffrement symétrique. `-d` et `-in` déchiffre le message.

`openssl list -cipher-commands` liste les algorithmes de chiffrement.

ssh-keygen

`ssh-keygen -t rsa -b 4096 -C "user@example.org"` génère une paire de clés (2 fichiers) contenant la clé privée et la clé publique.

Lors de l'utilisation de cette commande, une phrase de passe (*passphrase*) est demandée. C'est une bonne pratique d'entrer une *passphrase* pour déverrouiller la clé.

Afin de ne pas devoir entrer la *passphrase* à chaque connexion ssh à une machine ce qui ferait perdre l'avantage de ne pas entrer son mot de passe, l'utilitaire `ssh-agent` peut s'occuper de conserver cette clé la durée d'une session par exemple (voir).

`ssh-keygen -lf .ssh/id_ed25519` permet d'afficher le *fingerprint* de la clé.

Ceci peut servir à diffuser ce *fingerprint* sur un *autre canal* ce qui donnera la possibilité aux utilisateurices de vérifier l'authenticité du serveur.

ssh-copy-id

`ssh-copy-id -i ~/.ssh/id_rsa.pub user@host` tente de déposer la clé publique `id_rsa.pub` de l'utilisateurice `user` sur le serveur `host`.

ssh-agent

`ssh-agent` est un programme qui maintient les clés privées et est appelé lors d'une connexion SSH. C'est grâce à lui qu'il ne faut pas répéter une *passphrase* à chaque connexion SSH (mais bien à

chaque lancement de l'agent. Généralement à l'ouverture d'une session.)

Le lancement de `ssh-agent` est un peu particulier et se fait par `eval $(ssh-agent)` afin de le lancer. Le *pid* est placé dans une variable d'environnement (`SSH_AGENT_PID`). Une fois lancé, il est nécessaire de lui donner les clés utilisées *via* `ssh-add`.

L'option `-t` permet de limiter le temps pendant lequel, `ssh_agent` stocke une clé. Par défaut, c'est *forever*.

ssh-add

`ssh-add <key file>` ajoute une clé à *ssh-agent*. Il est peut-être nécessaire d'entrer une *passphrase*.

ssh

`ssh user@host.domainname` tente une connexion ssh à la machine distante.

Voici quelques options (consulter la page de manuel pour la documentation complète) :

- `-i <identity file>` renseigne une clé ssh. Par défaut les noms de fichiers habituels sont tentés (voir *man pages*).
- `-J <destination>` se connecte à la machine *destination* avant de se connecter à la destination finale (à partir de la machine *destination*). Permet donc de faire un saut. `--l <login>` utilise ce login au lieu de celui de l'utilisateur. Identique à faire précéder l'arobase `@` du login. `-p <port>` renseigne le numéro de port. Par défaut, c'est le port 22.

`ssh -Q key` donne la liste des algorithmes de cryptographie asymétrique.

`ssh -Q kex` donne la liste des algorithmes cryptographiques d'échanges de clés.

`ssh -Q cipher` donne la liste des algorithmes de cryptographie symétriques.

Pour la liste complète des options de requête de l'option `-Q`, consulter la page de manuel.

knock

`knock <host> <ports>` voir page 46.

Monitoring et fichiers de logs

Les fichiers de logs 49
Pour aller plus loin 50
Le coin des commandes 50
who 50
ps 50
last 50
cat 50

Un élément de sécurité système est la surveillance (*monitoring*) du réseau. Il est important de surveiller sa machine ou le réseau pour détecter les activités non autorisées et les trous de sécurité.

On peut chercher:

- les permissions qui auraient été modifiées sur certains fichiers[32];
- les derniers *users* connectés (Quand ? D'où ?) *via* la commande `last`
- … consulter les fichiers de logs.

Les fichiers de logs

Le système et les services tournant fournissent des informations au système sur leur fonctionnement ; les activités et les évènements qu'ils produisent. Ces informations incluent, outre les activités normales, les avertissement, les erreurs, les performances et toutes informations utiles pour le dépannage et la surveillance du système.

Ces fichiers de *logs*, en français, les *fichiers journaux*, se trouvent dans `/var/log`.

Le fichier principal est `/var/log/syslog` et beaucoup de services offrent le ou les leurs.

Par exemple `/var/log/apache2/access.log` et `/var/log/apache2/errors.log` pour le service *apache2*.

Une manière simple de consulter ces fichiers sont les utilitaires `cat`, `grep` (avec l'option `-r`, voir page 43) et `tail` (voir page 43).

Dans un environnement plus professionnel, il sera nécessaire de *faire remonter* les logs et de les centraliser à un endroit où ils pourront être lus. En effet, une équipe d'administrateurs et administratrices système ne va pas les consulter individuellement sur chaque machine qu'elle gère. Les logs peuvent être cen-

[32] Particulièrement des fichiers sensibles comme `/etc/passwd`, ceux lancés par `cron`, des exécutables… ce qui devient compliqué à « faire à la main ».

tralisés.

Une manière de faire remonter ces logs est d'utiliser la pile `telegraf` - `InfluxDB` - `Grafana` :

- `telegraf` est un utilitaire capable de reporter des métriques dans une base de données *InfluxDB*;
- `InfluxDB` est une base de données conçues pour enregistrer des métriques, des évènements systèmes… Elle peut recevoir les valeurs de plusieurs agent `telegraf`, éventuellement, répartis sur plusieurs machines;
- `Grafana` est un service web capable d'afficher des données en provenance d'une base de données *InfluxDB* par exemple.

Pour aller plus loin

L'étape suivante est l'étude des IDS (*Intrusion Detection Système*), systèmes de détection d'intrusion… mais c'est une autre histoire…

Le coin des commandes

who

`who` affiche les *users* actuellement connectés à la machine ; à partir de quel terminal, depuis quand et avec quelle IP.

ps

`ps` liste les processus en cours d'exécution (voir page 17)

last

`last` donne les dernières connexions au système (la commande utilise *wtmp* qui logue ces connexions dans `/var/log/wtmp` *binary file*). *wtmp* logue les connexions au système avec l'utilisateurice, l'IP, la date et les moments — début, fin — de la connexion.

```
last
last root | grep -v console
```

Extrait d'une commande `last`

```
alice    pts/6        192.168.210.8    Tue Jan 17 20:18 - 21:45  (01:27)
```

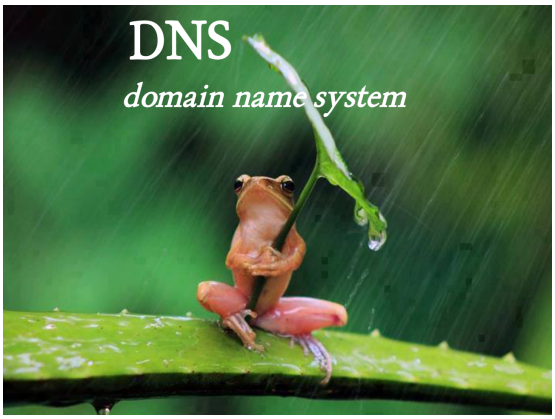
cat

`cat` affiche le contenu d'un fichier sur la sortie standard. Cette commande intervient régulièrement dans un *pipe*. On peut lui préférer la commande `bat` qui est un *clone* de la commande `cat` avec coloration syntaxique et intégration de *git* (option `-d`).

DNS domain name system

La résolution de noms **51**
1984, 2000, 2014 53
Fonctionnement d'un serveur DNS **54**
Fonctionnement d'une requête 55
Une question de confiance 57
DNSSEC **60**
Les différents champs et les fichiers de zone **61**
Serveurs DNS `bind9` et `unbound` **64**
`bind9` 64
`unbound` 65
La configuration du *stub resolver*, le fichier `resolv.conf` **66**
`DNS` menteur ou *response policy zone* (mais c'est moins vendeur) . **67**
`bind` 67
`unbound` 68
Le coin des commandes **69**
dig 69
Localiser une adresse IP 71

There is no place like 127.0.0.1… perhaps ::1



Les adresses IP n'étant pas conviviales, nous retenons les noms de machines… un serveur de noms permet de faire la correspondance entre un nom d'hôte et une adresse IP.

La résolution de noms

Au commencement, les machines étaient peu nombreuses et rarement connectées à internet. La correspondance entre les noms de machines et les adresses IP se faisait dans le fichier `/etc/hosts`.

Un fichier *hosts* a l'allure suivante [33] :

[33] Souvent un fichier *hosts* ne résout plus que la boucle locale et, éventuellement, des noms à des fins de tests ou de développement.

```
$ cat /etc/hosts
127.0.0.1 localhost
127.0.0.1 harmony.in.esigoto.info \
    harmony
[cut]
```

Avec l'agrandissement des réseaux locaux, et l'augmentation des tables d'hôtes à maintenir sur ces machines, un service s'est chargé de recopier certains fichiers de configuration entre machines. Il s'agit de `NIS` (*network information server*). Les fichiers sont maintenus sur une seule machine, le serveur, et les clients interrogent ce serveur plutôt que de lire la version locale du fichier.

Nous sommes dans les années 80, `NIS`[34] centralise et facilite la maintenance du fichier `/etc/hosts` mais également d'autres : `/etc/passwd`, `/etc/group`…

Plus tard, à partir de 1994, `DNS` lui sera préféré. En plus de mémoriser les noms de machines locales, `DNS` permet de faire la recherche pour tous les noms internet. Il permet la gestion d'un grand nombre de noms pas son système de dissémination de l'information et de mise à jour.

L'ordre dans lequel le système fait la résolution de noms se trouve dans le fichier `/etc/nsswitch.conf` et plus particulièrement avec l'entrée `hosts` qui peut avoir l'allure suivante par exemple :

```
hosts: files mdns4_minimal \
    [NOTFOUND=return] dns
```

- dans ce cas, le nom d'hôte sera recherché dans le fichier *hosts* puis *via multicast dns* et enfin, par une requête auprès du serveur DNS.

1984, 2000, 2014

Ces années marquent au sujet des noms de domaines.

Pour rappel, un nom de domaine est de la forme `example.org`. Une machine de ce domaine portera par exemple le nom `harmony.example.org`.

Les noms de domaine ont une structure hiérarchique. Chaque partie, appelée label, est séparée par un point. La partie la plus à droite est le domaine de premier niveau (*tld top level domain*) et doit être choisie parmi les *tld* existants.
Par exemple `org`.

La deuxième partie doit comporter entre 1 et 63 caractères et pour certains *tld* peut être accentuée.
Par exemple `example`.

Ces deux parties forment ce que l'on appelle habituellement le nom de domaine.
Par exemple `example.org`.

À ces noms de domaine peuvent être ajoutés d'autres label pour former des sous-domaines.
Par exemple `foo.example.org`, `bar.example.org`.

Au commencement toujours, les noms de domaine de premier niveau génériques (*gtld*) sont : `com`, `net` et `org` rapidement suivi de `int`, `edu`, `gov` et `mil`. Il s'agit des années 1984 et 1985.

Les années 2000 voient fleurir quelques nouveaux noms de domaines : `biz`, `info`, `aero`, `coop`, `museum`, `name`, `pro`, `jobs`, `travel`, `cat`, `mobi`, `asia`, `tel`, `xxx`, `post` et `sexy`. Il est encore possible de les recenser.

[34] `NIS` s'appelle d'abord *yellow pages* et, bien après avoir été renommé, certaines commandes commencent encore par `yp`.

À ces noms de domaines s'ajoutent les noms de domaines de premier niveau nationaux (*cctld*) comme `be` pour la belgique.

Puis en 2014, c'est l'explosion, toute personne désirant un nom de domaine de premier niveau peut le demander et, moyennant finance, il sera disponible. À l'heure où je rédige, je compte 1551[35] nouveaux domaines de premiers niveaux (*new gtld*) là où ils étaient 22 dans les années 80.

La liste complète des noms de domaines de premiers niveaux se trouve sur le site de l'IANA *Root zone database* (s.d.).

Fonctionnement d'un serveur DNS

> Serveur DNS ? Serveur DNS faisant autorité ou résolveur ?

Lors de l'interrogation d'un serveur DNS, soit le serveur connait la réponse à la question et la donne, soit il la cherche.

Le DNS peut connaitre la réponse à la question parce que celle-ci se trouve dans son **cache** ou parce que le serveur **a autorité pour la zone** concernée. Une information a une certaine durée de vie (*TTL*, *time to live*) déterminée par le serveur ayant autorité. Un serveur ayant autorité pour une zone (un domaine de premier niveau, un domaine, un sous-domaine) détient la liste des correspondances IP/nom et nom/IP pour la zone. Il peut communiquer l'information au serveur qui la demande. Si le serveur ne connait pas la réponse, il la cherche.

Il existe différents types de serveurs DNS :

- les **résolveurs** (*resolver*)[36] ou serveur DNS « à cache seul »,

  ces serveurs ont une bonne mémoire puisqu'ils ne peuvent répondre qu'avec les informations détenues en cache.

  S'ils n'ont pas l'information en cache, ils interrogent un serveur racine (voir ci-dessous) s'ils sont récursifs (*recursive*) ou un autre résolveur appelé *for-*

[35] Les noms des domaines de premier niveau commençant par `a` : .aaa .aarp .abarth .abb .abbott .abbvie .abc .able .abogado .abudhabi .ac .academy .accenture .accountant .accountants .aco .active .actor .ad .adac .ads .adult .ae .aeg .aero .aetna .af .afamilycompany .afl .africa .ag .agakhan .agency .ai .aig .aigo .airbus .airforce .airtel .akdn .al .alfaromeo .alibaba .alipay .allfinanz .allstate .ally .alsace .alstom .am .amazon .americanexpress .americanfamily .amex .amfam .amica .amsterdam .an .analytics .android .anquan .anz .ao .aol .apartments .app .apple .aq .aquarelle .ar .arab .aramco .archi .army .arpa .art .arte .as .asda .asia .associates .at .athleta .attorney .au .auction .audi .audible .audio .auspost .author .auto .autos .avianca .aw .aws .ax .axa .az

[36] Le terme résolveur peut prêter à confusion. Il ne faudra pas confondre le résolveur dans le sens de la partie du serveur DNS faisant la résolution de nom pour remplir son cache — le résolveur — et la configuration du résolveur du système qui précise quel est le serveur DNS à utiliser — le *stub* résolveur. Cette dernière configuration étant généralement faites dans le fichier `/etc/resolv.conf`. Nous y reviendrons.

warder s'ils ne le sont pas.

- les serveurs « faisant autorité » (*authoritative server*) pour une ou plusieurs zones,

  ces serveurs détiennent un fichier par zone[37] contenant les correspondances entre les adresses IP et les noms.

  Parmi ces serveurs ayant autorité, certains seront des **serveurs maitres** — détenant effectivement les fichiers de zone — et d'autres seront des **serveurs esclaves** qui obtiennent leurs fichiers d'un serveur maitre.

Ces deux aspects sont très différents et doivent être bien compris. Là où le résolveur donne la réponse contenue dans son cache, le serveur ayant autorité donne la réponse qu'il connait. La réponse qui se trouve dans un fichier de configuration ou autre. Les données ayant autorités doivent toujours être prioritaires sur les données contenues dans le cache. Si les deux services sont séparés, lors de l'interrogation d'un résolver, le client sait que la réponse n'a pas autorité et qu'elle a une durée de vie limitée. Elle n'est peut-être plus valable et il faudra un peu de temps — dépendant de la durée de validité de la donnée — pour obtenir la « bonne » valeur. Lors de l'interrogation d'un serveur ayant autorité, le client sait que l'information reçue est toute fraiche.

Dans ces notes, nous traitons avec `bind` qui est capable d'assumer convenablement les deux rôles.

Fonctionnement d'une requête

Un serveur DNS résolveur récursif a un fonctionnement **top/down** avec **cache**.

Détaillons le fonctionnement d'une requête DNS faites par un résolveur DNS récursif. Lorsqu'il ne connait pas la réponse à la question posée, il la cherche. Pour ce faire, le serveur interroge l'un des **serveurs racines** (*root servers*) — au hasard dans la liste qu'il détient. Aujourd'hui, la liste des serveurs racines est la suivante (extrait) :

```
$ cut /etc/bind/db.root
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.cache
;           on server           FTP.INTERNIC.NET
;       -OR-                    RS.INTERNIC.NET
;
;       last update:    February 17, 2016
```

[37] Ils en auront généralement deux par zone ; un pour la zone et la résolution nom → IP et un second pour la *zone inverse* et la résolution IP → nom.

```
;       related version of root zone:   2016021701
;
; formerly NS.INTERNIC.NET
;
.                       3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.     3600000      AAAA  2001:503:ba3e::2:30
;
;; --cut--
;
; OPERATED BY ICANN
;
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     199.7.83.42
L.ROOT-SERVERS.NET.     3600000      AAAA  2001:500:3::42
;
; OPERATED BY WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
M.ROOT-SERVERS.NET.     3600000      AAAA  2001:dc3::35
; End of file
```

Le serveur racine (*root server*) interrogé répond en renseignant l'IP — plutôt *une* IP — du serveur ayant autorité pour la zone de premier niveau concernée. Le serveur DNS requérant interroge alors cette nouvelle IP. Si le serveur interrogé a autorité, il répond, sinon, il renseigne le serveur ayant autorité pour le sous-domaine… et ainsi de suite jusqu'à la réponse. Une fois la réponse obtenue, le serveur DNS requérant conserve la réponse en cache pendant sa durée de vie (*TTL*).

Par exemple, pour une requête `pica.esigoto.info` faite *via* `dig +trace pica.esigoto.info` (extraits) et en s'aidant du schéma 2 page 58 :

- demande au serveur DNS qui répond avec la liste des serveurs racines ;

```
.  3600000 IN NS D.ROOT-SERVERS.NET.
.  3600000 IN NS L.ROOT-SERVERS.NET.
.  3600000 IN NS F.ROOT-SERVERS.NET.
.  3600000 IN NS K.ROOT-SERVERS.NET.
.  3600000 IN NS M.ROOT-SERVERS.NET.
.  3600000 IN NS I.ROOT-SERVERS.NET.
.  3600000 IN NS E.ROOT-SERVERS.NET.
.  3600000 IN NS H.ROOT-SERVERS.NET.
.  3600000 IN NS J.ROOT-SERVERS.NET.
.  3600000 IN NS C.ROOT-SERVERS.NET.
.  3600000 IN NS B.ROOT-SERVERS.NET.
.  3600000 IN NS A.ROOT-SERVERS.NET.
.  3600000 IN NS G.ROOT-SERVERS.NET.
```

- demande à `D.ROOT-SERVERS.NET` qui répond avec la liste des serveurs ayant autorité pour la zone `info` ;

```
info. 172800 IN NS a0.info\
```

```
    .afilias-nst.info.
info. 172800 IN NS a2.info\
    .afilias-nst.info.
info. 172800 IN NS b0.info\
    .afilias-nst.org.
info. 172800 IN NS b2.info\
    .afilias-nst.org.
info. 172800 IN NS c0.info\
    .afilias-nst.info.
info. 172800 IN NS d0.info\
    .afilias-nst.org.
```

- demande à `b2.info.afilias-nst.org` qui répond avec la liste des serveurs ayant autorité pour la zone `esigoto.info` ;

```
esigoto.info. 86400 IN NS \
    ns-4-c.gandi.net.
esigoto.info. 86400 IN NS \
    ns-167-b.gandi.net.
esigoto.info. 86400 IN NS \
    ns-83-a.gandi.net.
```

- demande à `ns-4-c.gandi.net` qui répond avec l'adresse IP de l'hôte *pica* car il a autorité pour la zone `esigoto.info` et connait donc la réponse ;

```
pica.esigoto.info.  1800  IN  \
    A  84.198.168.131
```

- la requête est bien *top/down* et le serveur requérant conservera en cache l'information pendant sa durée de validité.

Un résolveur *forwarder* quant-à lui fera la requête auprès de son *forwarder* qui répondra si la réponse est dans son cache ou qui exécutera la requête sinon… *via* un autre *forwarder* ou de manière récursive en fonction de sa configuration.

Remarque Avant novembre 2021, la question complète était transmise aux différents serveurs. Le serveur racine recevait, par exemple, `pica.esigoto.info` à résoudre alors que seule la partie `info` lui est utile. Ce fonctionnement viole le principe de minimisation des données traitées mis en avant par le RGPD (Règlement Général pour la Protection des Données).

La RFC9156[38] solutionne ce problème en proposant *Query Name Minimisation to Improve Privacy* ou, plus simplement, *QNAME minimisation*. Avec cette approche, seule la partie utile du nom est transmise.

Lire à ce sujet : « *RFC 9156: DNS Query Name Minimisation to Improve Privacy* » Bortzmeyer (s.d.)

Une question de confiance

Il y a deux aspects au sujet de la notion de **confiance** dans le service DNS :

- la confiance dans le serveur lui-même ;
- la confiance dans le réseau.

Confiance dans le serveur DNS Éliminons de suite la confiance dans le logiciel lui-même qui dépendra bien sûr du logiciel choisi : *bind* dans ces notes et c'est pareil pour *unbound*. Pour le logiciel libre, à l'habitude, le code peut-être audité et il est donc possible de vérifier

[38] http://www.rfc-editor.org/info/rfc9156

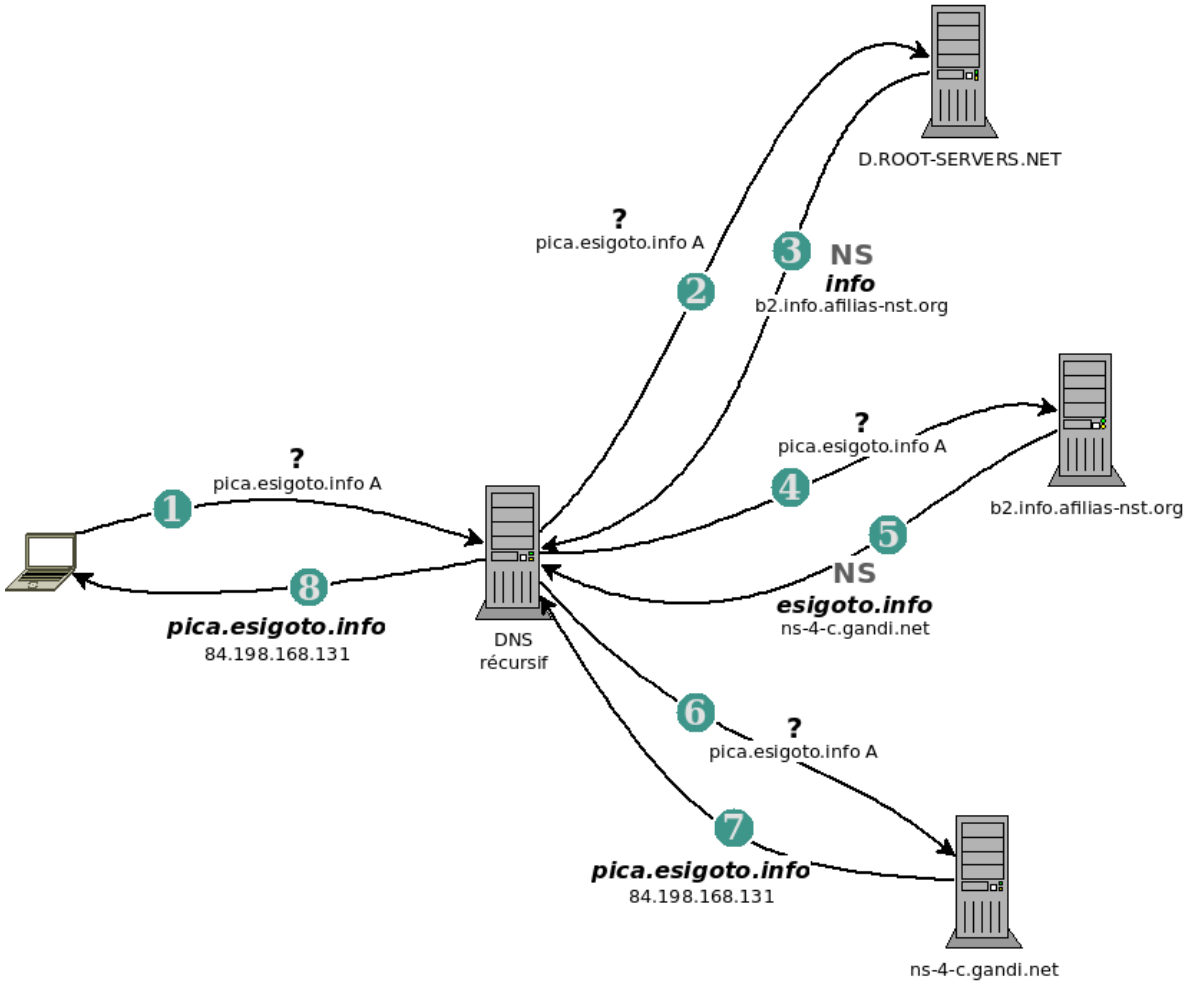


FIGURE 2 – Illustration d'une requête DNS récursive

que le logiciel fait bien ce qu'il prétend. Accordons notre confiance à *bind* et à *unbound*.

Bien que quasi personne ne choisisse quel serveur il utilise, ce devrait être fait car un serveur DNS peut **mentir** ou **bloquer** certaines requêtes. Un serveur DNS peut mentir pour gagner de l'argent, pour respecter les lois — ou se plier aux contraintes c'est selon — d'un état…

Lire à ce sujet :
« Mise en place d'un DNS menteur » Bettens (s.d.)
« Quand ton serveur DNS te bloque ou te ment » Bettens (2021)

Lorsque l'on ne choisit pas son résolveur DNS, c'est le serveur DHCP qui le propose et, dans un réseau local, c'est la *box* qui sera résolveur DNS. Ce résolveur est généralement *forwarder* et le *forwarder*, est le résolveur DNS du fournisseur d'accès à internet (*FAI*).

Dans ces conditions quelle confiance accorde-t-on à son FAI ?
Quelles sont les raisons qui pourraient le faire mentir ou le faire bloquer des requêtes ?

Il est alors possible de choisir un résolveur public. Les plus connus aujourd'hui sont ceux de Google `8.8.8.8` et de Cloudfare `1.1.1.1`. Ces serveurs n'utilisent (probablement[39]) pas `bind`. Ils s'annoncent plus rapides puisqu'ils font plus rarement de requêtes récursives auprès des serveurs racines, ont de bonnes connections réseaux…

Ces serveurs me bloquent-ils ?
Me mentent-ils ?
Quelles sont les informations, ou plutôt quelle quantité d'informations récoltent-ils ?
Quid de ma vie privée ?
Que reste-t-il de la décentralisation du service ?

Il est également possible, d'installer son propre résolveur. Pour sa machine ou pour son réseau local ou d'entreprise. Notons que si beaucoup de personnes font ce choix, cela peut avoir un impact sur les serveurs racines dont la charge risque d'augmenter.

Bref, tout n'est pas blanc ou noir…

Confiance dans le réseau Ceci étant dit, un autre aspect de la confiance dans ce service est notre intimité et les requêtes DNS circulent par défaut en clair sur le réseau. Je peux me connecter en `https` au site `example.org` et personne ne saura ce que je consulte hormis le propriétaire du site mais *tout le monde* saura que j'y accède puisque ma demande de résolution de nom circulera en clair.

Si mon résolveur est sur ma machine, pas de problème, seul l'administrateurice aura accès aux requêtes qui sont effectuées… mais pas que. Le résolveur se trouvant sur ma machine va sortir ses requêtes en clair et une personne à l'écoute de mon réseau pourrait les voir.

Ne parlons pas de l'utilisation d'un serveur public. Le chemin est long entre ma machine et `1.1.1.1` par exemple.

Deux solutions existent pour remédier à cet aspect :

[39] Probablement pas puisque ces services ne documentent pas vraiment.

- **DNS sur TLS** (*dot*) encapsule les requêtes DNS dans une connexion chiffrée avec TLS ;
- **DNS sur HTTPS** (*doh*) encapsule les requêtes DNS dans une connexion chiffrée avec HTTPS.

L'avantage de la seconde sur la première est aussi un désavantage, elle permet l'utilisation de `HTTPS` qui est généralement disponible *un peu partout*. Les ports sont généralement laissés ouverts. Pour le reste, les concepts sont identiques.

Les serveurs publics tels que *Cloudflare* par exemple proposent ces services.

doh et *dot* peuvent être mise en œuvre à l'aide de `dnsdist`… ce qui sort du cadre de ces notes.

DNSSEC

DNSSEC est une extension de sécurité à DNS. Les enregistrements fournis par DNS sont **chiffrés** et les zones **signées**.

Les données proviennent bien initialement du serveur ayant autorité pour la zone et n'ont pas été altérées en chemin puisqu'elles ont été signées par le propriétaire de la zone et que cette signature peut être vérifiée.

DNSSEC apporte l'**authenticité** et l'**intégrité** des données. Il apporte aussi un **chainage de confiance**.

Chaque **zone** a une paire de clés publique-privées : la `ZSK` (*zone signed key*). Cette clé est utilisée pour signer les données de la zone. La partie publique de cette clé est disponible dans un enregistrement DNS : `DNSKEY`.

Pour un label donné (par exemple `host.example.org`), les enregistrements d'un même type sont rassemblés dans un **RRset** et c'est ce RRset qui sera signé. Voir figure 3 page suivante.

Un enregistrement donné est donc accompagné de sa signature. Par exemple :

```
example.org.4214 IN A 93.184.216.34
example.org.4214 IN RRSIG A 13 2
  86400 (
    20231013015700 20230922122122
    64700 example.org
  6gn68jzj2mdS[cut]Z3D0BmQvQ== )
```

- `A`, le type d'enregistrement
- 13, l'algorithme utilisé
- 2, le nombre de labels du RRset
- 86400, le TTL originel
- 20231013015700, la date d'expiration
- 20230922122122, ?
- 64700, *keytag* (identifiant non-unique de la clé)
- example.org, nom du signataire
- …, la signature proprement dite.

Le RRset, l'enregistrement RRSIG et la clé DNSKEY peuvent ensemble valider la réponse donnée par le serveur DNS. Voir figure 4 page 62

La clé de zone (ZSK) est utilisée pour signer chaque RRset. Pour authentifier la clé de zone, elle est signée par la clé de signature de clé `KSK` (*key signing key*). Pour garantir l'authenticité de cette clé,

elle est *hashée* et le hash est maintenu dans l'enregistrement DS de la la zone parent. De parent en parent s'établit la **chaine de confiance**.

La zone `.` signe `.org.` qui signe `example.org.`.

Lorsqu'un enregistrement n'existe pas le serveur DNS ne donne pas de réponse. Avec DNSSEC, il est possible de garantir l'absence de réponse. DNSSEC donne alors un enregistrement NSEC (obsolète) ou NSEC3 garantissant que l'enregistrement n'existe pas. NSEC[3] signe la réponse vide.

NSEC est réputé sensible à *zone walking attack*. En demandant si un enregistrement existe, NSEC répond en donnant l'enregistrement ce qui garanti que l'enregistrement demandé n'existe pas mais donne une information sur la zone. À partir de là, il est possible d'obtenir plusieurs nom dans la zone.

Par exemple demander l'enregistrement `a.dnstests.ovh` donne un enregistrement NSEC qui précise que `sub.dnstests.ovh` existe.

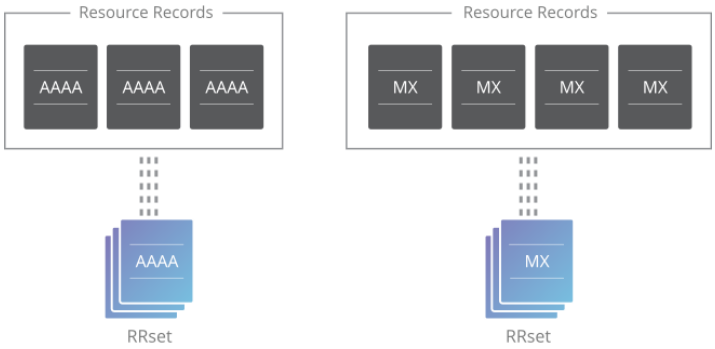


FIGURE 3 – Exemples de RRset (source Cloudflare)

Les différents champs et les fichiers de zone

La requête DNS par défaut est celle demandant l'adresse IP correspondant au nom, c'est une requête pour l'enregistrement `A`. Il existe différents types d'enregistrements disponibles sur le site de l'IANA *Domain Name System (DNS) Parameters* (s.d.). Voici les principaux :

- `A` adresse IPv4 (*host address*)
- `AAAA` adresse IPv6 (*host IPv6 address*)
- `NS` serveur de nom ayant autorité (*authoritative name server*)
- `CNAME` alias (*canonical name for an alias*)
- `SOA` début du fichiers de zone (*start of a zone of authority*)
- `PTR` nom de domaine (*domain name pointer*)
- `MX` serveur de mail (*mail exchange*)
- `TXT` zone de texte (*text strings*)
- `DNSKEY` TODO (*dnskey*)
- `AXFR` transfert de zone (*transfer of an*

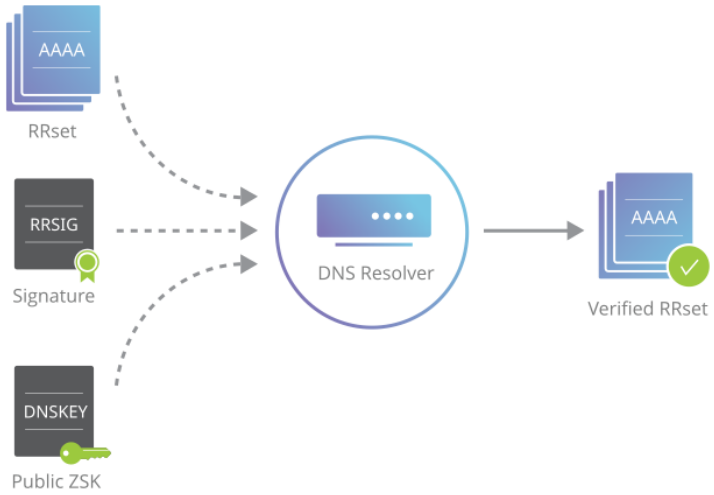


FIGURE 4 – Validation d'un enregistrement (source Cloudflare)

entire zone)

DNSSEC ajoute les champs suivants :

- `DNSKEY` partie publique de la clé de zone ZSK
- `RRSIG` signature cryptographique du `RRset` correspondant
- `DS` le hash de la DNSKEY
- `NSEC` déni d'existence explicite d'un enregistrement (obsolète)
- `NSEC3` déni d'existence explicite d'un enregistrement
- `CDNSKEY` et `CDS` pour une zone enfant demandant des mises à jour dans la zone parent.

Ce sont ces différents types d'enregistrements que contiennent les fichiers de zone. Un fichier de zone contient :

- l'enregistrement `SOA` précisant la zone, il est requis et c'est le premier enregistrement ;
- un enregistrement `NS` précisant le serveur de nom ayant autorité pour la zone, il est requis ;
- des enregistrements `MX` précisant le serveur de mail. Idéalement ils sont au moins deux ;
- des enregistrements `A`, `AAAA` et `CNAME` pour les zones ;
- des enregistrements `PTR` pour les zones inverses ;
- des enregistrements `TXT`,

  ces enregistrements contiennent différentes informations. Par exemple, des commentaires pour la validation d'un certificat *https*, des valeurs `SPF`, `DMARK`… pour la sécurisation et la paramétrisation des serveurs de mails…

La zone locale a cette allure :

```
$TTL    604800
@ IN SOA localhost. root.localhost. (
         2         ; Serial
```

```
    604800         ; Refresh
     86400         ; Retry
    419200         ; Expire
    604800 )       ; Negative Cache TTL
;
  IN NS      localhost.
@ IN A       127.0.0.1
@ IN AAAA    ::1
```

Comme illustration d'une zone plus générale, la zone `esigoto.info` ayant quelques services en IPv4 et en IPv6 pourrait avoir cette allure :

```
@ 10800 SOA ns1.gandi.net. hostmaster.gandi.net. (
    2021030501 2h 30m 30d 1h
)
@ 10800 IN MX 10 spool.mail.gandi.net.
@ 10800 IN MX 50 fb.mail.gandi.net.
@ 10800 IN A 91.121.216.124
@ 10800 IN AAAA 2001:41d0:8:52c9:0:ff:fee3:bd57

atacama 1800 IN A 84.198.168.129
atacama 1800 IN TXT "v=spf1 ip4:84.198.168.129 mx -all"

pica 1800 IN A 84.198.168.131
pica 1800 IN TXT "v=spf1 ip4:84.198.168.131 mx -all"
vlab 1800 IN CNAME pica
vlabesi 1800 IN CNAME pica

date 10800 IN CNAME momos.hipocoon.be.
imap 10800 IN CNAME access.mail.gandi.net.
paste 10800 IN A 91.121.216.124
paste 10800 IN AAAA 2001:41d0:8:52c9:0:ff:fee3:bd57
smtp 10800 IN CNAME relay.mail.gandi.net.
www 10800 IN A 91.121.216.124
www 10800 IN AAAA 2001:41d0:8:52c9:0:ff:fee3:bd57
```

- les commentaires du champ `SOA` ne sont pas présents. Les commentaires ne sont pas obligatoires ;
- les valeurs ne sont pas toutes données en secondes mais avec une unité. Par exemple `h` pour les heures ;
- deux champs `MX` sont proposés et une priorité est donnée aux serveurs de mails : l'une de 10 et l'autre de 50 ;

Serveurs DNS bind9 et unbound

bind (*Berkeley Internet Name Daemon*) est sans nul doute l'implémentation la plus répandue du DNS. La première version date de 1984 et bind9 est une réécriture des versions plus anciennes qui prend en charge *DNSSEC* par exemple.

bind9 peut jouer le rôle de résolveur **et** de serveur DNS ayant autorité.

unbound quant-à lui est un résolveur DNS plus récent (2004) pouvant avantageusement remplacer bind lorsqu'il n'est pas nécessaire d'avoir aussi un serveur DNS ayant autorité.

bind9

L'installation de bind9 est aussi simple que :

```
># apt install bind9
```

Dès le paquet installé, le serveur est lancé et jouera le rôle de *resolver*.

Les fichiers de configuration de bind9 se trouvent dans /etc/bind/. Le répertoire a l'allure suivante (*debian buster*) :

```
$ tree /etc/bind
/etc/bind
├── bind.keys
├── db.0
├── db.127
├── db.255
├── db.local
├── db.root
├── named.conf
├── named.conf.default-zones
├── named.conf.local
├── named.conf.options
├── rndc.key
└── zones.rfc1918
```

Les fichiers db.* définissent les zones (cfr. ci-dessus). Le choix du nom de ces fichiers est libre.

named.conf est le point d'entrée de la configuration de *bind*. Chez *debian*, il contient des *includes* vers les fichiers named.conf.* :

- named.conf.options contient les options dans la section options { } dédiée.

  Principalement les interfaces sur lesquels le serveur va répondre. Par défaut le serveur DNS est local ;

- named.conf.local contient les zones pour lesquels le serveur a autorité (*master* et *slave*). Chacune dans une section *zone*, par exemple zone "example.org" { } ;

- named.conf.default-zones contient les zones par défaut : le *broadcast* db.0, la zone inverse pour la boucle locale db.127, la zone inverse pour le *broadcast* db.255, la boucle locale db.local, la zone . db.root[40] ;

- bind.keys contient la clé *DNSSEC* pour les serveurs racines.

- rndc.key contient le *hash* de la clé

[40] Dans les paquets récents de *bind*, ce n'est plus le fichier db.root qui est utilisé, mais le fichier /usr/share/dns/root.hints. C'est une bonne pratique de vérifier — à partir de named.conf quels fichiers de configuration sont utilisés. Ceci peut varier d'une distribution à l'autre.

d'accès au serveur bind *via* rndc ;

- `zones.rfc1918` définit toutes les zones correspondant aux classes d'adresses privées comme étant vides.

Une fois la configuration faite, le service se gère à l'aide de *systemd via* la commande `systemctl` comme habituellement.

Au sujet des *logs*, ils se trouvent dans `/etc/syslog` dès lors que *bind* est configuré pour parler. Ceci peut se faire en ajoutant une section *logging* au fichier de configuration. Cette section peut avoir l'allure suivante (très verbeuse)_:

```
logging {
  category default { default_syslog;
    default_debug; };
  category security { default_syslog;
     default_debug; };
  category database { default_syslog;
     default_debug; };
  category resolver { default_syslog;
     default_debug; };
  category queries { default_syslog;
    default_debug; };
  category unmatched { null ; };
};
```

`unbound`

L'installation de `unbound` est tout aussi simple et le serveur est également lancé à l'installation.

Les fichiers de configuration se trouvent dans `/etc/unbound` et le répertoire à cette allure :

```
$ tree /etc/unbound
/etc/unbound/
├── unbound.conf
├── unbound.conf.d
    ├── qname-minimisation.conf
    └── root-auto-trust-anchor \
      -file.conf
```

`unbound.conf` contient la configuration du serveur… qui peut se résumer à inclure les fichiers contenus dans `unbound.conf.d`.

Quelques éléments de configuration d'unbound :

```
server:
    # can be uncommented if you do not need user privilige protection
    # username: ""

    # can be uncommented if you do not need file access protection
    # chroot: ""

    # location of the trust anchor file that enables DNSSEC. note that
    # the location of this file can be elsewhere
    auto-trust-anchor-file: "/usr/local/etc/unbound/root.key"
```

```
# send minimal amount of information to upstream servers to enhance privacy
qname-minimisation: yes

# specify the interface to answer queries from by ip-address.
interface: 0.0.0.0
# interface: ::0

# addresses from the IP range that are allowed to connect to the resolver
access-control: 192.168.0.0/16 allow
# access-control: 2001:DB8/64 allow
```

Les paramètres `local-zone` et `local-data` (voir Labs (s.d.)) permettent de définir une zone locale. Par exemple :

```
private-domain : "local"
local-zone : "local" static
local-data : "harmony.local. \
  IN A 10.1.0.42"
```

La documentation complète est disponible « Unbound by NLnet Labs » Labs (s.d.)

La configuration du *stub resolver*, le fichier `resolv.conf`

Le résolveur — ou *stub resolver* — est un ensemble de routines de la librairie C qui donne accès au DNS. Les programmes susceptibles de faire appel à ces routines sont nombreux. Il s'agit de tous les programmes nécessitant une résolution de noms. Par exemple : `ssh`, `git`, `owncloud`, `dropbox`… mais surtout les navigateurs[41].

Le *stub resolver* se configure dans le fichier `/etc/resolv.conf` qui aura au minimum l'allure suivante :

```
nameserver <IP>
```

où `IP` représente l'adresse IP du serveur DNS résolveur à interroger.

À cette ligne peuvent s'ajouter d'autres informations comme le domaine (*domain*) dans lequel se trouve la machine, des noms de domaines dans lesquels chercher un nom d'hôte (*search*), un

[41] Les navigateurs sont configurés par défaut pour utiliser le serveur DNS défini par le système. Pour un réseau local, ce serveur DNS est (très souvent) celui de la *box* (routeur-modem donnant accès à internet) et se réfère au serveur DNS du fournisseur d'accès à interenet (*FAI*). Les navigateurs modernes peuvent être configurés pour faire leurs requêtes DNS *via* HTTPS (*doh*). Dans ce cas, une configuration habituelle est d'utiliser un serveur DNS chez *Cloudflare*. Il en existe d'autres. Il est possible d'utiliser son serveur DNS local bien sûr. Dans un réseau d'entreprise, ce sont les administrateurices systèmes qui déterminent où sont faites les requêtes DNS.

timeout éventuel, un nombre de tentatives (*attempts*)… Habituellement, un fichier `resolv.conf` dans une configuration familiale aura l'allure suivante :

```
search domainname
nameserver 192.168.1.1
```

le serveur DNS résolveur utilisé est celui de la *box*.

Pour utiliser un serveur DNS résolveur local et demander au *stub resolver* d'ajouter un nom de domaine aux noms « sans point », un fichier `resolv.conf` pourrait avoir cette allure :

```
search in.esigoto.info
nameserver ::1
nameserver 127.0.0.1
```

Remarque Il serait facile de croire que la modification du fichier `resolv.conf` suffit à modifier le serveur DNS résolveur. C'est exact lorsqu'il n'y a pas un programme qui met ce fichier à jour. Lorsque l'on utilise un serveur DHCP, le système reçoit le nom du résolveur du serveur DHCP… ce qui écrase les valeurs écrites « à la main » dans *resolv.conf*.

Dans ce cas, une manière de faire est de ne pas demander l'adresse du résolveur au serveur DHCP. Ceci peut être fait en **retirant le mot** `domain-name-servers` du fichier `/etc/dhcp/dhclient.conf`.

Il est également souvent possible de paramétrer le client DHCP *via* le *GUI* (*Graphical User Interface*, l'outil graphique). Par exemple *NetworkManager*, peut se configurer pour ne demander au serveur DHCP que les paramètres réseaux (adresse, *mask* et *gateway*) et de fixer soi-même l'adresse du DNS à utiliser.

`DNS` menteur ou *response policy zone* (mais c'est moins vendeur)

Un serveur DNS peut mentir ou bloquer des requêtes. Si ton DNS te bloque, tu croiras que le site n'existe pas. C'est de la censure. Si ton DNS te ment, tu seras dirigé vers une autre page.

Faire mentir, c'est mettre en place en place **RPZ** *Response Policy Zone*.

Bortzmeyer[42] et Paul Vixie[43] expliquent le principe qui est décrit dans la documentation de bind9[44] §6.2.16.20 p98- et dans celle d'unbound[45].

`bind`

Pour indiquer à bind que l'on veut utiliser la RPZ (*response policy zone*) il faut l'ajouter dans les options;

```
response-policy {
  zone "liar.local";};
```

Cette zone peut être une zone définie ailleurs et contenant une liste de sites à bloquer ou bien je peux la définir moi-même…

[42] http://www.bortzmeyer.org/rpz-faire-mentir-resolveur-dns.html
[43] https://www.dns-oarc.net/files/workshop-201103/rpz2.pdf
[44] http://www.bind9.net/arm910.pdf
[45] https://unbound.docs.nlnetlabs.nl/en/latest

Ajouter:

```
zone "liar.local" {
    type master;
    file "/etc/bind/ \
    db.local.liar.local";
  allow-query {none;};
};
```

Ajouter le fichier de zone. Dans ce fichier :

- pour qu'un nom soit renseigné comme inexistant (*undefined*),

```
<name> CNAME .
```

- pour qu'un nom soit renseigné comme vide (*empty set of resources*),

```
<name> CNAME *.
```

- pour remplacer l'IP,

```
<name> A <IP>
<name> AAAA <IP>
```

dans l'exemple suivant;

```
$TTL 1h
@ SOA eve.liar.local.
    root.localhost. (
    2017030302
    2h 30m 30d 1h
)
  NS eve.liar.local.

eve IN  A   127.0.0.1

example.be CNAME .
example.org A 127.0.0.1
example.com CNAME   eve.liar.local.
```

- `example.be` ne répondra pas, le nom n'étant pas défini ;
- `example.org` répondra avec l'adresse de la boucle locale ;
- `example.com` sera redirigé vers `eve.liar.local`.

`unbound`

Dans le cas d'unbound, il est nécessaire de charger le module `respip` et de définir une section `rpz`.

```
server:
  module-config: "respip […]"
rpz:
  [cut]
```

Voir la documentation officielle[46].

[46] https://unbound.docs.nlnetlabs.nl/en/latest/topics/filtering/rpz.html

Le coin des commandes

dig

`dig` est une commande permettant d'interroger un serveur DNS.

À chaque réponse sont associés des drapeaux (*flags*) en voici quelques uns :

- `AA` *Authoritative Answer* (RFC1035)
- `TC` *Truncated Response* (RFC1035)
- `RD` *Recursion Desired* (RFC1035)
- `RA` *Recursion Available* (RFC1035)
- `CD` *Checking Disabled* (voir DNSSEC)
- `AD` *Authentic Data* (voir DNSSEC)

Requête simple auprès du résolveur configuré par défaut pour obtenir l'adresse IP correspondant au nom :

```
$ dig esigoto.info

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> esigoto.info
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10754
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2e6152d9a4eee435fad46701604b6c9bc051247825ee77d9 (good)
;; QUESTION SECTION:
;esigoto.info.          IN  A

;; ANSWER SECTION:
esigoto.info.       10741   IN  A   91.121.216.124

;; AUTHORITY SECTION:
esigoto.info.       78196   IN  NS  ns-4-c.gandi.net.
esigoto.info.       78196   IN  NS  ns-167-b.gandi.net.
esigoto.info.       78196   IN  NS  ns-83-a.gandi.net.

;; ADDITIONAL SECTION:
ns-4-c.gandi.net.   570 IN  A   217.70.187.5
ns-4-c.gandi.net.   570 IN  AAAA    2604:3400:aaac::5

;; Query time: 8 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: ven mar 12 14:28:59 CET 2021
;; MSG SIZE  rcvd: 204
```

Le résolveur interrogé n'a pas autorité pour la zone.

- les *flags* `ra` et `rd` signifie qu'il y a eu récursion ;
- la valeur du *ttl*, ici 78196, n'est pas « entière » et une prochaine requête identique verrait décroitre cette valeur.

La même requête auprès d'un serveur DNS ayant autorité donnerait :

```
$ dig esigoto.info @ns-4-c.gandi.net

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> esigoto.info @ns-4-c.gandi.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7687
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;esigoto.info.          IN  A

;; ANSWER SECTION:
esigoto.info.       10800   IN  A   91.121.216.124

;; Query time: 22 msec
;; SERVER: 2604:3400:aaac::5#53(2604:3400:aaac::5)
;; WHEN: ven mar 12 14:41:36 CET 2021
;; MSG SIZE  rcvd: 57
```

Le serveur ayant autorité pour la zone,

- le *flag* `aa` est levé et signifie que c'est une réponse d'autorité ;
- la valeur du *ttl*, ici 10800, est telle que définie dans le fichier de zone ;

Demande à un serveur racine la liste des serveurs racines (sans paramètre, c'est le comportement par défaut de `dig`) :

```
dig . NS @a.root-servers.net
```

L'option *trace* demande de montrer la requête récursive : `dig +trace example.org`.

L'option *search* demande d'ajouter le nom de domaine à la requête : `dig +search pica`.

L'option *short* donne une réponse (très) courte.

Il est possible d'utiliser l'enregistrement (*record*) `AXFR` pour initier un transfert de zone. C'est une faille de sécurité que d'autoriser ce transfert de zone. Cette

commande devrait donc échouer[47].

```
dig axfr example.org
```

`dig <name> DNSKEY +multi` donne l'enregistrement `ZSK` de la zone.

`dig ds example.org [+dnssec] +multi` donne le hash de la clé de zone. Cette valeur est donnée par la zone `.org.`.

`dig A +dnssec a.dnstests.ovh` confirme l'absence de réponse — le champ est vide — avec un enregistrement NSEC.

`dig A +dnssec public.example.org` confirme l'absence de réponse — le champ est vide — avec un enregistrement NSEC3.

Localiser une adresse IP

Il peut être intéressant de savoir à quelle région est attribuée une adresse IP, voire à quelle fournisseur d'accès (*FAI*). Pour ce faire, il faut utiliser un service tiers, par exemple ipapi.co[49]. Dans une console, une requête peut avoir cette allure :

```
$ curl https://ipapi.co/<IP>/yaml
$ curl https://ipapi.co/<IP>/yaml -s \
    | awk '/country\_name|region|org/'
```

1. version longue au format `yaml`;
2. version réduite au pays, à la région et à l'organisation éventuelle.

[47] N'étant pas le seul à vouloir illustrer le transfert de zone, Robin des Bois[48] (*sic*) met une zone à disposition autorisant ce transfert de zone. À l'heure ou j'écris `dig axfr @nsztm1.digi.ninja zonetransfer.me` fonctionne.

[49] http://ipapi.co

SAMBA ou l'intégration de machines MS Windows et GNU Linux

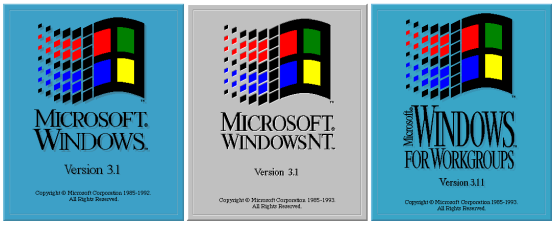
Partager pour travailler ensemble… 72
Des protocoles 74
Installation 77
Les dæmons 77
Configuration de Samba 78
Variables 79
Configuration de la liste d'exploration 80
Authentification des utilisateurices 80
Le coin des commandes 82
`smbclient` 82
`testparm` 82
`samba-tool` 82
`nmbloockup` 82

Partager pour travailler ensemble...

Fin des années 80, nous sommes à l'époque où l'on commence à relier les ordinateurs entre eux. C'est l'époque des cables coaxiaux sans IP qui relient les ordinateurs entre eux. Les personnes qui utilisent ses machines veulent qu'elles puissent communiquer pour se partager des fichiers… tout en étant authentifiées.

Ceci est rendu possible par **Microsoft Windows 3.11 *for worksgroup*** dans le monde Microsoft et son protocole *NetBIOS* (voir ci-dessous) qui permettra :

- le **partage des fichiers** entre utilisateurs et utilisatrices de machines différentes ;
- l'**authentification** mutuelle pour que chaque personne accède aux fichiers pour lesquels elle a les droits.



À sa suite, quelques années plus tard, *LAN Manager* se chargera de sécuriser l'accès aux données avec :

- l'**identification** permettant d'établir l'identité du l'utilisateurice ;
- l'**authentification** permettant de vérifier cette identité. *Es-tu bien qui tu prétends être ?* et ;
- l'**autorisation** pour donner, ou non, l'accès aux ressources.

LAN Manager connaitra quelques évolutions (voir ci-dessous) pour évoluer vers *Kerberos*.

Le projet *Samba* propose une suite d'outils permettant la communication — l'interopérabilité — entre machines MS Windows et **nix*. Ces outils reposent sur le protocole *SMB* (*Server Message Block*). Ce protocole, natif sur MS Windows, permet :

- le partage de fichiers dépendant de la version du protocole sur MS Windows et de Samba sur **nix* ;
- le partage d'imprimantes ;
- la gestion centralisée des groupes ;
- la gestion centralisée des droits d'accès aux fichiers, répertoires et imprimantes ;
- la gestion centralisée de l'identification et de l'authentification *à la mode* « PDC NT4 » (*Primary Domain Controller NT4*) dans un premier temps. Depuis toujours, Samba, a la volonté d'en faire un véritable *Active Directory*.

Un ***Active Directory*** est une suite de programmes pour centraliser, gérer et authentifier les utilisateurs et utilisatrices ainsi que les ordinateurs d'un domaine afin de sécuriser les données, et contrôler les accès. L'**AD** (*Active Directory*) maintient un état des lieux du domaine afin de permettre la gestion des authentifications, des droits et des accès aux ressources du domaine.

Différentes version de *Samba* se succèdent au fil des ans :

- **Samba 1**, implémente *LAN Manager* et supporte la notion de *workgroup* (groupe de travail) amenée par MS Windows 3.11 *for workgroups*

- **Samba 2**, amène le support de contrôleur de domaine au sens *PDC NT4* ;

- **Samba 3**, améliore le support de *PDC NT4* et des nouvelles versions de SMB ;

- **Samba 4** (*Samba 4.0.0*[50]) est une réécriture de Samba commencée en 2005 et pleinement fonctionnelle à partir de 2012, pour que Samba puisse prendre en charge complètement la fonctionnalité d'*Active Directory* (au sens Microsoft). Samba s'appuie sur les spécifications officielles de MS

[50] https://www.samba.org/samba/news/releases/4.0.0.html

Windows qui sont (ou ont été[51]) souvent mal documentées et sur du *reverse engineering* pour proposer une implémentation fonctionnelle d'*active directory*.

À partir de *Samba 4.2.0*, le support de *Samba 3* sera abandonné (mais pas le support de l'authentification et l'identification NT4).

Samba 4 comprend un server LDAP, un serveur d'authentification Kerberos, un server DNS dynamique[53] et implémente tous les appels de procédure nécessaires à *Active Directory*.

Samba supporte également Python 3.

Des protocoles

NetBIOS (*Network Basic Input Output System*) est une version étendue de BIOS, prenant en charge les échanges de données sur un réseau local grâce à un protocole de transport approprié.

Il y a deux implémentations de *NetBIOS*, l'une, **NBT** s'appuyant sur TCP/IP est aujourd'hui la norme tandis que l'autre, **NetBEUI** (*NetBIOS Extended User Interface*) n'est plus utilisée.

L'interface NetBIOS consiste en un ensemble de services permettant d'identifier et de gérer des connexions entre systèmes. Ces différents services sont :

- le service de noms ; noms de groupe de travail (*workgroup*) et noms de machine ;
- le service de sessions ;
- le service de datagrammes pour l'envoi d'informations sans connexion.

C'est un mode de nommage en couche 4, sur les ports 137, 138 et 139.

Un nom NetBIOS se compose de 16 caractères : 15 pour le nom et le 16^{e} pour le rôle. Un machine déclare au minimum deux noms : le nom de machine et celui du *workgroup*.

Voici la liste des caractères associés aux différents rôles :

[51] Ironiquement, lors de la sortie de Samba 4, Microsoft se fend de félicitations précisant qu'elle (la société Microsoft) s'est engagé à soutenir l'interopérabilité entre les plateformes pour cet outil qu'est Active Directory. Elle se dit heureuse que la documentation fournie (entre autre) ait aidé Samba à développer la fonctionnalité d'Active Directory de Samba. Microsoft oublie que si cette documentation est maintenant disponible, c'est à cause de (grâce à ?) sa condamnation par la Commission Européenne lors d'un jugement anti-trust en 2004 et en 2007. L'accord trouvé entre Microsoft et PFIF (*Protocol Freedom Information Foundation*) — une organisation à but non lucratif agissant pour des projet comme le projet Samba — est de céder les informations techniques nécessaires pour la (modique) somme de 10 000 € (ou 14 400 $). Ça fait un peu « bal des faux-culs ! » (source[52]).

[53] Dans ce cas, il est configuré comme *forwarder via* la directive `dns forwarder = <IP>` dans `/etc/samba/smb.conf` où IP représente l'IP du serveur DNS vers lequel les requêtes externes sont *forwardées*.

Name	Number(h)	Type	Usage
`<computername>`	00	U	Workstation Service
`<computername>`	01	U	Messenger Service
`<\\--__MSBROWSE__>`	01	G	Master Browser
`<computername>`	03	U	Messenger Service
`<computername>`	06	U	RAS Server Service
`<computername>`	1F	U	NetDDE Service
`<computername>`	20	U	File Server Service
`<computername>`	21	U	RAS Client Service
`<computername>`	22	U	Microsoft Exchange Interchange(MSMail Connector)
`<computername>`	23	U	Microsoft Exchange Store
`<computername>`	24	U	Microsoft Exchange Directory
`<computername>`	30	U	Modem Sharing Server Service
`<computername>`	31	U	Modem Sharing Client Service
`<computername>`	43	U	SMS Clients Remote Control
`<computername>`	44	U	SMS Administrators Remote Control Tool
`<computername>`	45	U	SMS Clients Remote Chat
`<computername>`	46	U	SMS Clients Remote Transfer
`<computername>`	4C	U	DEC Pathworks TCPIP service on Windows NT
`<computername>`	42	U	mccaffee anti-virus
`<computername>`	52	U	DEC Pathworks TCPIP service on Windows NT
`<computername>`	87	U	Microsoft Exchange MTA
`<computername>`	6A	U	Microsoft Exchange IMC
`<computername>`	BE	U	Network Monitor Agent
`<computername>`	BF	U	Network Monitor Application
`<username>`	03	U	Messenger Service
`<domain>`	00	G	Domain Name
`<domain>`	1B	U	Domain Master Browser
`<domain>`	1C	G	Domain Controllers
`<domain>`	1D	U	Master Browser
`<domain>`	1E	G	Browser Service Elections
`<INet~Services>`	1C	G	IIS
`<IS~computer name>`	00	U	IIS
`<computername>`	[2B]	U	Lotus Notes Server Service

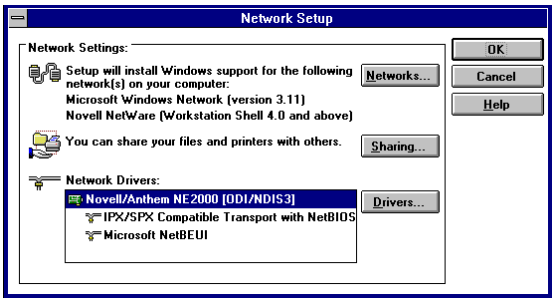
SMB (*Server Message Block*) prend en charge les services de niveau supérieur. Il se base sur des échanges de messages relatifs à la session, au contrôle des fichiers, à l'impression et à la communication.

Historiquement, SMB est apparu avec **Microsoft 3.1 *for workgroup*** en octobre 1992. C'est la première version de Microsoft Windows supportant le travail en groupe et le (fameux) *voisinage réseau*.

Le protocole SMB a évolué au fil des années et se décline en plusieurs versions : *SMBv1* (qui est par défaut désactivé dans les versions >4.10 de Samba), *SMBv3* et *SMBv3*.

Les notions de domaines et de groupes de travail (*workgroups*) se réfèrent à la notion de systèmes participant à un même réseau. Tous ces systèmes utilisent la même liste de machines ; la liste de *browsing*. Cette liste de *browsing* est fournie au client par le maitre d'exploration principal (*master browser*). Elle est associée à un groupe de travail et est utile pour limiter les *broadcasts* lors de la résolution des noms NetBIOS.

La liste de *browsing* est un service de la couche 2 : NetBIOS.



Un domaine ou un groupe de travail (*workgroup*) est le nom donné à un ensemble de machines. Ces machines pourront partager des ressources. Ce concept est différent du **domaine de sécurité MS Windows** géré par un contrôleur de domaine : anciennement un **contrôleur de domaine principal** (PDC *primary domain controller*) ou actuellement un ***active directory*** (AD).

WINS (*Microsoft Windows Internet Name Service*) est le service de nom internet de Microsoft Windows. Le serveur WINS maintient la liste des noms NetBios se trouvant dans le réseau. Dans le cas de NBT, il maintient la correspondance *nom NetBIOS / IP*. WINS permet d'éviter le *broadcast* sur le réseau ; au lieu d'un *broadcast* une machine se signale directement au serveur WINS. La présence d'un serveur WINS permet de « passer les routeurs »[54].

- un client (au *boot* ou au lancement de Samba) demande au serveur WINS d'ajouter son nom et son IP. La requête est acceptée si l'adresse n'est pas utilisée[55].
- lorsque le client (ou le service Samba) s'arrête, le système avise le serveur WINS qui rend le nom disponible.

Remarque : le protocole NetBIOS n'est pas *routable*, IP oui.

winbind (*Name Service Switch daemon for resolving names from NT servers*) est le nom associé au dæmon `winbindd` qui intègre le système d'authentification et les protocoles d'identification aux services AD (*user/group lookup*) disponibles sur un domaine Windows. Il est utile à Samba bien sûr et également à *nsswitch*

[54] Dès qu'une mécanisme arrête l'utilisation du *broadcast* pour directement contacter une adresse IP sur un certain port… ce mécanisme peut quitter le segment et être routé. C'est par exemple, le cas de la liste d'exploration.

[55] C'est bien le client qui informe le serveur WINS de l'IP qu'il possède déjà pour que le serveur la mémorise. Il ne s'agit **pas d'obtenir** une adresse IP. Un serveur WINS n'est pas un serveur DHCP.

(*Name Service Switch*) et à PAM.

Il n'est pas à proprement parlé utilisé par *nsswitch* mais fournit des services de gestion de connexions au contrôleur de domaine à `smbd`, `ntlm_auth` et au module PAM `pam_winbind`

nsswitch (*Name Service Switch*) permet d'obtenir de diverses bases de données (telles que NIS ou DNS) les informations systèmes ou concernant les utilisateurs et utilisatrices du système. Le comportement de *nsswitch* se configure dans `/etc/nsswitch.conf`. Le service *winbind* obtient le même genre d'information du contrôleur de domaine (au sens MS Windows NT). Ajouter *winbind* aux endroits que vont bien dans le fichier `nsswitch` demande de faire une requête auprès du contrôleur de domaine.

Un des travaux de `winbindd` est de conserver une table de correspondance entre les *users* et les *groups *nix* et ceux de Windows NT (auxquels sont assignés un identifiant unique SID (*security id*)).

Installation

L'installation n'appelle pas de commentaire. Samba est disponible dans les paquets de la distribution. Comme toujours, on privilégiera l'intallation *via* le gestionnaire de paquets plutôt qu'à partir des sources.

Les dæmons

Samba est associé aux dæmons :

- **smbd** (*smb dæmon*) pour le partage des systèmes de fichiers et des imprimantes, l'authentification et les droit d'accès ;
- **nmbd** (*name server NetBIOS dæmon*) est associé au service de noms WINS et prend en charge la résolution de noms. Il peut-être configuré pour être *master browser* par exemple…
- **windbind** (*Name Service Switch daemon for resolving names from NT servers*) est associé aux services liés au contrôleur de domaine ou à l'*active directory*.

Les services peuvent être gérés à l'aide de `systemctl` comme d'habitude.

Configuration de Samba

La configuration de Samba se fait dans un unique fichier `smb.conf` habituellement dans `/etc/samba/smb.conf`[56].

Ce fichier se décompose en **sections**. Chaque section débute par un nom, le nom de service, écrit entre crochet « `[]` ». La section se termine par le début de la suivante ou la fin de fichier. Chaque section correspond à un partage excepté la section `[global]` généralement en début de fichier.

Il existe trois sections particulières :

- `[global]` Définit les paramètres communs du serveur pour le partage de toutes les ressources. Les options définies dans cette section s'appliquent à tous les autres partages comme si le contenu de la section y était copié. Heureusement, une option définie dans `[global]` peut être reprise et modifiée dans une autre section.
- `[homes]` Permet à un ou une utilisateurice distante d'accéder à son répertoire *home* (`/home/alice`). Ce compte sera connecté à son répertoire si ce répertoire personnel existe, il doit donc posséder un compte sur la machine !

  Supposons qu'une machine cliente tente de se connecter à un partage *alice* sur un serveur. Aucun partage de ce nom n'est défini dans le fichier `smb.conf` mais il existe une section `[homes]`. Samba trouve le compte utilisateur *alice* dans la base des mots de passe, puis compare les mots de passe. Si les deux mots de passe concordent, Samba crée un partage `alice` pour le client, ce partage sera `/home/alice`.
- `[printers]` Cette section définit les paramètres pour le partage des imprimantes. Cette section permet d'éviter de définir un partage par imprimante. Samba vérifie dans le fichier `/etc/printcap` si c'est une imprimante et la met à disposition du client.

  Il est également possible de définir une section particulière (par exemple `[yaprinter]` définissant le partage de l'imprimante. Cette section contiendra le paramètre `printable = yes` indiquant que le partage se rapporte à une imprimante.

À la suite d'un nom de section, se trouve la liste des **paramètres**. Chaque ligne est de la forme :

```
param = value
```

Les lignes de **commentaires** commencent par un hashtag « `#` » ou par un point virgule « `;` ».

Le fichier de configuration de Samba est rechargé chaque minute. Il est possible de forcer la relecture du fichier de configuration de Samba en envoyant le signal `SIGHUP` au serveur comme ceci :

```
kill -SIGHUP <pid smbd>
kill -SIGHUP <pid nmbd>
```

ou plus simplement en utilisant `systemctl` et

[56] Pour demander à `smbd` quel fichier de configuration il lit : `smbd -b | grep smb.conf`

```
systemctl reload smbd nmbd
```

Variables

Le fichier de configuration peut contenir des variables fixant les caractéristiques du serveur Samba et des clients qui s'y connectent. Ces variables sont de la forme `%A`.

Cas d'usage : supposons que le réseau comporte plusieurs clients susceptibles de se connecter au serveur mais que l'utilisateur `bob` nécessite une configuration particulière pour la section `[homes]`. En écrivant cette configuration dans `/etc/samba/smb-bob.conf`, le fichier *smb.conf* pourra avoir cette allure :

```
[homes]
    ; …
    include = /etc/samba/smb-%m.conf
    ; …
```

Dans ce cas si le fichier existe, l'*include* se fera sinon, aucune erreur ne sera générée (ce qui est pratique pour toutes les autres connexions).

Voici la liste des variables issue de Samba Eckstein *et al.* (2000).

Variable	**Définition**
Variables clients	
`%a`	Architecture du client (Samba, MS Windows…)
`%I`	Adresse IP du client
`%m`	Nom NetBIOS du client
`%M`	Nom DNS du client
Variables utilisateur	
`%u`	Nom d'utilisateur Unix actuel
`%U`	Nom d'utilisateur client demandé (parfois inusité par Samba)
`%g`	Groupe principal de `%u`
`%G`	Groupe principal de `%U`
`%H`	Répertoire de base de `%u`
Variable de partage	
`%p`	Chemin d'accès du montage automatique associé au répertoire racine du partage (si différent de `%P`)
`%P`	Répertoire racine du partage en cours
`%S`	Nom du partage en cours
Variables serveur	
`%d`	ID de processus du serveur en cours
`%h`	Nom d'hôte du serveur Samba
`%L`	Nom NetBIOS du serveur Samba
`%N`	Serveur de répertoire de base établi à partir de la table d'automount

Variable	Définition
`%v`	Version de Samba
Autres variables	
`%R`	Niveau de protocole SMB pris en compte dans la négociation
`%T`	Date et heure courantes

Configuration de la liste d'exploration

L'exploration (*browsing*) désigne la fonction permettant d'examiner les serveurs et les partages disponibles sur un réseau. Cette liste est visible dans le *voisinage réseau*.

C'est le client qui informe par *broadcast* le maitre explorateur (*master browser*) ou directement le serveur WINS de sa présence par une annonce `__MS_BROWSE__`.

L'option `browsable` permet de montrer ou cacher un partage tout comme l'ajout d'un dollar « `$` » en début de nom de partage permet de le faire avec MS Windows.

Un des machines du réseau, celle ayant le rôle `1D` d'explorateur local de domaine (*local master browser*), est responsable de la conservation, mise à jour et diffusion de cette liste. D'autres options interviennent dans l'élection de sa *master browser* que nous ne détaillons pas ici. Il s'agit des options : *os level*, *local master*, *netbios name*, *server string*, *preferred master*…

Authentification des utilisateurices

Le paramètre `security` définit le type d'authentification pour le serveur. Ce paramètre peut prendre les valeurs `user`, `ads` ou `domain` (voire `auto`). La valeur par défaut est `user`[57].

- `user` Le client doit se connecter avec des identifiants valides avant tout.

  Le nom de la ressource à laquelle le client veut se connecter n'est pas transmis au serveur tant que le client n'est pas authentifié. C'est pourquoi pour qu'un partage *guest ok* ne sera

[57] Les anciennes versions de Samba acceptaient les valeurs `share` et `server` qui ne sont plus d'actualité.

pas accessible sans authentification excepté si le serveur associe les utilisateurs inconnus au compte *guest*. Cfr. `map to guest`.

- `domain` Dans ce mode, le client doit rejoindre le domaine de sécurité MS Widows au sens *MS Windows NT*.
- `ads` Dans ce mode, le client doit rejoindre le domaine de sécurité MS Windows au sens *membre d'un active directory* (AD). Dans ce mode, Kerberos doit être installé et configuré.

Les utilisateurices et leur mot de passe et éventuellement les groupes doivent être stockés quelque part. Le paramètre `passdb backend` définit la manière dont ils le seront.

Les différents *backend* sont :

- `smbpasswd` l'ancien fichier plat ;
- `tdbsam` la base de données triviale (*trivial data base* **TDB**) est une base de données très simple ;
- `ldapsam` l'url vers un annuaire LDAP.

  L'url est optionnelle. Si elle est absente, ce sera `ldap://localhost`.

```
passdb backend = tdbsam
passdb backend = tdbsam:/etc/samba\
    /private/passbd.tdb
passdb backend = ldapsam:\
    ldaps://ladp.example.org
```

Pour l'échange de secret entre le client et le serveur, Samba utilise un protocole de type *challenge / response*.

L'idée étant que le client puisse convaincre le serveur qu'il connait le secret sans que celui-ci ne transite sur le canal non sûr.

1. le client envoie un premier message *negotiate_message* proposant différents protocoles ;
2. le client répond avec un *challenge_message* ;
3. le client peut calculer un hash *authenticate_message* basé sur le *challenge_message* et sur le mot de passe ;
4. après vérification, le serveur accept ou non.

Ce protocole a évolué au gré des évolutions des algorithmes de hashage pour passer de DES à MD5… Dans sa première version, Microsoft NT Lan Manager (*ntlm*) utilise DES et ne devrait plus être utilisé aujourd'hui.

- `ntlm v1` le *challenge* est de 8 bytes (64 bits), les *hash* font 16 bytes et sont
  - `LanMan`, un DES du message. Le message est limité à 14 caractères ASCII ou
  - `NT`, un MD4 encodés cette fois en UTF16
- `ntlm v2` à partir de *Microsoft Windows NT4 service pack 4*. Le *hash* à une longueur de plus de 24 bytes et la fonction de hashage est MD4, MD5, HMAC MD5…

Au niveau des systèmes Microsoft Windows : LanMan est utilisé par les versions de MS Windows inférieures à MS Windows NT (*aka* MS Windows 95-98) et *ntlm v2* n'est disponible qu'à partir de MS Windows NT4 SP4. `ntlm v1` est désactivé par défaut.

Il existe également plusieurs version du protocole SMB.

- `SMBv1` Déprécié en 2013 et désactivé par défaut à partir de Samba 4 (et

MS Windows Server 2016).

- `SMBv2`
- `SMBv3`

Le coin des commandes

`smbclient`

`smbclient` est à l'origine l'outil de test d'une installation de Samba. *smbclient* peut faire des transferts de fichiers « à la FTP », imprimer, archiver (`tar`), envoyer des messages, rechercher les services…

```
smbclient -L <netbios name>
smbclient -L <hostname>
smbclient -L <netbios| host name> \
    -U <user>

smbclient -L harmony -U alice
```

- liste les partages du serveur (`harmony`) pour le *user* `alice`

```
smbclient //<netbios|host name>\
    /<share name>

smbclient //harmony/distri
```

- se connecte en *ftp-like* au partage (`distri`) du serveur (`harmony`)

`testparm`

`testparm` vérifie la validité syntaxique du fichier *smb.conf*. Il retourne la valeur des différents paramètres.

```
testparm smb.conf
```

`samba-tool`

`samba-tool` est l'outil d'administration de Samba (en particulier lorsqu'il joue le rôle de contrôleur de domaine).

```
samba-tool domain provision
  --realm=<kerberos domain uppercase>
  --domain <domain name>
  --server-role=dc
```

- configure Samba avec le rôle de contrôleur de domaine en utilisant Kerberos

```
samba-tool user setpassword
  administrator
```

- modification du mot de passe du compte « administrator »

`nmbloockup`

`nmbloockup` recherche les noms NetBIOS et les associe avec une adresse IP.

```
nmbloockup [-A] <IP | netbios name>

nmbloockup harmony
nmbloockup -A 127.0.0.1
```

PAM, Pluggable Authentication Modules

Du côté de l'administrateurice 83
Du côté du développeur ou de la développeuse 85

Ensemble de bibliothèques responsables de la centralisation de l'**authentification** Linux.

Une application *pam enabled* est une application déléguant l'authentification aux modules PAM. Lors de son lancement, une telle application interroge l'API PAM pour qu'elle exécute différentes tâches. Ces tâches sont décrites dans un fichier de configuration. Si toutes ces tâches sont vérifiées, l'authentification est réussie et l'application continue, sinon l'application prend fin.

Les applications *pam enabled* les plus connues sont : cron, login, passwd, su, sudo…

Avant l'arrivée de PAM chaque application était responsable de la gestion de l'authentification et devait se mettre à jour lorsque le système changeait de *backend* ou pour permettre un autre type d'authentification. Par exemple, je peux vérifier qu'un ou une utilisatrice est bien qui elle prétend être en vérifiant dans le fichier `/etc/passwd` ou, plus tard, dans le fichier `/etc/shadow` ou encore, si l'administratueurice le désire dans un annuaire LDAP… Il est difficile que toutes les applications gèrent tous les systèmes d'authentification passés… et à venir. PAM résout ce problème.

La liste des modules disponibles se trouve dans `/etc/security` (pour *debian*, c'est dans `/lib/x86_64-linux-gnu/-security/`) et la documentation *via* `man pam_<module>` par exemple `man pam_mail`.

Du côté de l'administrateurice

La configuration se fait dans le répertoire `/etc/pam.d`[58] dans lequel se trouve un fichier par application et un fichier de configuration par défaut pour les applications qui n'auraient pas de configuration spécifique.

Un fichier de configuration *pam* se compose de ligne de la forme :

```
module-type control-flag module-path
args
```

- `module-type` peut prendre les valeurs : `auth`, `account`, `session` et `password`
  - `account` gère le compte pour

[58] Il est possible de trouver un fichier `/etc/pam.conf` vestige des temps anciens. Si vous lisez de la documentation conseillant d'écrire dedans ou (pire) de le créer, c'est une veille doc.

tout ce qui ne relève pas de l'authentification. Par exemple, permettre ou restreindre l'accès en fonction de l'heure, limiter les ressources ou encore, le terminal à partir duquel le compte se connecte ;

– `auth` gère l'authentification. Premièrement est-ce que l'utilisateurice est bien qui iel prétend être (connaissance du mot de passe) et deuxièmement la gestion des groupes ;

– `password` gère la mise à jour du mot de passe (pour chaque type d'authentification) ;

– `session` gère les actions qui peuvent être faites avant ou après la connexion de l'utilisateurice (*log*, *mount*…).

- `control-flag` peut prendre les valeurs : `required`, `requisite`, `sufficient` et `optional`[59]

  `-required` l'échec du module induit l'échec de l'authentification après l'exécution de la pile de modules.

  `-requisite` l'échec du module induit l'échec de l'authentification immédiatement après l'exécution du module (et pas à la fin de la pile).

  `-sufficient` la réussite de ce module (et de tous les modules requis précédents) induit la réussite de l'authentification. L'échec de ce module est ignoré et induit le passage aux modules suivants sur la pile.

  `-optional` l'échec ou la réussite de ce module n'a de l'importance que s'il est le seul de la pile de ce type (`module-type`).

- `module-path` est le chemin complet du module qui devra être exécuté ;

- `args` sont les arguments éventuels a passer au module.

Exemple illustré :

```
1 auth required
2     /lib/security/pam_securetty.so
3 auth required
4     /lib/security/pam_env.so
5 auth sufficient
6     /lib/security/pam_ldap.so
7 auth required
8     /lib/security/pam_unix.so
9     try_first_pass
```

1. concerne l'authentification et est requis. Il s'agit du terminal à partir duquel le compte essaie de se connecter.
2. concerne l'authentification et est requis. Ce module permet de positionner des variables d'environnement.
3. concerne l'authentification et est suffisant, sa réussite termine la pile d'appel. Ce module demande une authentification auprès d'un annuaire LDAP.
4. concerne l'authentification et est requis. Il ne sera exécuté que si le précédent échoue. Il utilisera le même mot de passe que celui ren-

[59] La partie *control* peut également prendre les valeurs `include` et `substack` qui ne seront pas traitées dans ces notes. Il existe également une syntaxe plus compliquée et plus récente qui ne sera pas non plus traitée dans ces notes. Elle est de la forme `[value1=action1 value2=action2…]`. Par exemple, l'équivalent de `required` dans cette syntaxe est `[success=ok new_authtok_reqd=ok ignore=ignore default=bad]`. Pour plus d'information : `man pam.conf`.

seigné à l'étape précédente.

Cette pile d'appel vérifie donc à partir de quel terminal se connecte l'utilisateurice, fixe des variables d'environnement, essaie une connexion LDAP et, en cas d'échec de cette identification LDAP fait une identification Linux classique.

Du côté du développeur ou de la développeuse

Pour écrire une application *pam-enabled* c'est-à-dire déléguant l'authentification aux modules PAM, il suffit de :

- inclure les librairies PAM comme *headers* ;

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
```

- faire un appel à PAM dans le code source (extraits) ;

```
int retval = pam_start(…);
if (retval == PAM_SUCCESS){
    // …pam start OK, now auth
    retval = pam_authenticate(…);
}
if (retval == PAM_SUCCESS){
    // …auth success,
    // is access permit ?
    retval = pam_acct_mgmt(…);
}
if (retval == PAM_SUCCESS){
    // …now all authorization ok
    // do stuff
    pam_end(…);
} else {
    // nope. not authorized
    exit(1);
}
```

- compiler en liant la librairie

```
cc -o yaapp yaapp.c \
    -lpam -lapm_misc -ldl
```

- ensuite, *root* choisira les modules que devra vérifier l'utilisateurice de l'application « `yaapp` ».

LDAP, Lightweight directory access protocol

Modèle d'information **87**
Modèle de nommage **88**
Format d'échange de données LDIF **89**
Modèle fonctionnel **90**
Implémentation du protocole : `OpenLDAP` 90
Schémas 91
ACL 92
Recherche 93
Le coin des commandes **93**

FIXME trouver une intro

LDAP définit un protocol d'accès à un **annuaire**. Un annuaire est une base de données spécialisée et avec une structure forte. Les accès en lecture sont réputés rapides. La recherche est efficace. Un annuaire n'est pas un système de gestion de base de données (*SGBD*). Un annuaire peut contenir tout ce qui peut être nommé. L'usage le plus fréquent est d'y stocker les utilisateurices de l'entreprise.

Un annuaire est destiné à être interrogé par des clients différents.

LDAP fournit un protocole d'échange entre les clients LDAP et un serveur LDAP. Les services offerts par un annuaire sont :

- un modèle d'information ;

  Ce modèle fournit les structures et types des données nécessaires à la construction de l'arbre de l'annuaire LDAP.

- un modèle de nommage ;

  Ce modèle définit comment les entrées et les données de l'arbre sont définies de manière unique.

- un modèle fonctionnel ;

  Ce modèle fonctionnel est le protocole LDAP en lui-même. Il définit le moyen d'accéder aux données dans l'arbre de l'annuaire. Les accès sont la connexion, l'authentification, la recherche, la lecture, la modification…

- un modèle de sécurité et de duplication.

Modèle d'information

Un annuaire à une **structure en arbre**, la racine contient la description de l'arbre, chaque nœud de l'arbre est une entrée et chaque nœud est un objet.

Généralement le premier niveau est une décomposition en unité organisationnelle (*organizational unit*). Par exemple, le département ressources humaines, informatique, ventes…

Un objet a un *nom*, un *identifiant* et des attributs obligatoires et optionnels.

objectClass
name, *objectId*, *attributes* (*must* ou *may*) et *type*

L'identifiant d'objet (*objectId*) est normalisé (par la RFC 2256). Le numéro représentant l'*objectId* respecte une certaine hiérarchie et un numéro peut être attribué à une société sur simple demande. Le numéro attribué à l'ESI est `1.3.6.1.4.1.23162` (cfr IANA enterprise numbers[60]). À partir de ce numéro, l'entreprise peut créer les identifiants qu'elle désire. Par exemple `1.3.6.1.4.1.23162.1`, `1.3.6.1.4.1.23162.2`…

La structure de ce numéro est la suivante. Pour l'exemple on ajoute un numéro pour *le local 504* puisque en-dessous du numéro attribué l'entreprise est libre de s'organiser comme elle l'entend :

```
iso(1)
|- org(3)
|--- dod(6)
|---- internet (1)
|----- private (4)
|------- enterprise (1)
|-------- esi.be (23162)
|--------- local504 (504)
```

Une classe d'objet peut par exemple, représenter une personne, sa définition est alors la suivante :

```
objectclass ( 2.5.6.6 NAME 'person'
    DESC 'RFC2256: a person'
    SUP top STRUCTURAL
    MUST ( sn $ cn )
    MAY ( userPassword
    $ telephoneNumber
    $ seeAlso
    $ description ) )
```

- le nom est `person` ;
- l'*objectId* est `2.5.6.6` ;
- l'objet est structurel et n'a pas de parent ;
- les attributs `sn` (*surname*) et `cn` (*common name*) sont obligatoires (*MUST*), les autres facultatifs (*MAY*)

Si cette classe, n'offre pas suffisamment d'attributs, il est possible d'utiliser une classe enfant, par exemple :

```
objectclass ( 2.5.6.7
  NAME 'organizationalPerson'
 DESC 'RFC2256: an organizational person'
  SUP person STRUCTURAL
  MAY ( title $ x121Address $
    registeredAddress $
    destinationIndicator $
    preferredDeliveryMethod $
    telexNumber $
    teletexTerminalIdentifier $
    telephoneNumber $
    internationaliSDNNumber $
    facsimileTelephoneNumber $
    street $ postOfficeBox $
```

[60] https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers

```
    postalCode $ postalAddress $
    physicalDeliveryOfficeName $
    ou $ st $ l )
)
```

- le nom est `organizationalPerson` ;
- l'*objectId* est `2.5.6.7` ;
- l'objet est structurel et son parent est `person`… il hérite donc de ses attributs et peut les utiliser ;
- les attributs `sn` et `cn` sont toujours obligatoires, les autres facultatifs.

Si cette classe n'offre pas suffisamment d'attributs, il est possible d'utiliser une classe enfant, par exemple :

```
objectclass ( 2.16.840.1.113730.3.2.2
  NAME 'inetOrgPerson'
  DESC 'RFC2798: Internet Org Person'
  SUP organizationalPerson
  STRUCTURAL
  MAY (
    audio $ businessCategory $
    carLicense $ departmentNumber $
    displayName $ employeeNumber $
    employeeType $ givenName $
    homePhone $ homePostalAddress $
    initials $ jpegPhoto $
    labeledURI $ mail $ manager $
    mobile $ o $ pager $
    photo $ roomNumber $ secretary
    $ uid $ userCertificate $
    x500uniqueIdentifier $
    preferredLanguage $
    userSMIMECertificate $ userPKCS12
    )
)
```

Et ainsi de suite.

Si aucune classe prédéfinie ne convient, il est possible de définir ses propres classes.

Les attributs sont également définis.

attributeType
name, *attributeId*, *description*

```
attributetype ( 2.5.4.4
  NAME ( 'sn' 'surname' )
  DESC 'RFC2256: last (family) name(s)
  for which the entity is known by'
  SUP name
)
```

Ces définitions sont disponibles en ligne (site de l'IANA) ou dans les fichiers `etc/ldap/schema/*`. Elles sont définies dans des **schémas**. À chaque insertion, le serveur vérifie si l'entrée est conforme au schéma. C'est le *schema checking*.

Modèle de nommage

Chaque nœud a un **identifiant unique** composé des attributs obligatoires pour la classe d'objet utilisée. C'est le ***distinguished name*** (**DN**). Par exemple :

```
uid=fpignon, ou=construction,
  dc=example, dc=org
```

Les deux parties `dc` sont les composants du nom de domaine qui forment le nom de la racine de l'annuaire. Pour la société `example.org`, la racine sera `dc=example, dc=org`.

L'attribut `dc` pour *domain component* est défini dans le schema *core.schema* :

```
attributetype ( 0.9.2342.19200300.\
```

```
100.1.25                                    SUBSTR caseIgnoreIA5SubstringsMatch
  NAME ( 'dc' 'domainComponent' )           SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 DESC 'RFC1274/2247: domain component'       SINGLE-VALUE
  EQUALITY caseIgnoreIA5Match              )
```

Format d'échange de données LDIF

LDIF *LDAP Data Interchange Format* est le format permettant l'échange des données entre une application et un annuaire LDAP ou entre annuaires et la modification des enregistrements.

Le format complet est décrit dans RFC 2849 et la base dans les pages de manuel `man ldif`.

Remarque Les espaces et les passages de ligne ont une certaine importance :

- chaque ligne est terminée par un line feed (`<LF>`) ou carriage return suivi de line feed (`<CR><LF>`) ;
- une ligne commençant par un *hash* « # » est un commentaire ;
- une ligne commençant par un espace () continue la ligne qui précède ;
- une ligne vide permet de séparer deux entrées différentes tandis que ;
- une ligne commençant par un tiret (*dash*) « - » permet de terminer une opération et permet d'en commencer une nouvelle (sur la même entrée)

Une entrée est de la forme :

```
dn: <distinguished name>
<attrdesc>: <attrvalue>
<attrdesc>: <attrvalue>
…
```

Un nœud de l'annuaire pourrait ressembler à :

```
dn:cn=Marlène Sassœur, ou=student,
 dc=example,dc=be
objectclass: inetOrgPerson
cn: Marlene SASSOEUR
sn: Marlene
mail: marlene.sassoeur@dev.null
description: Elle me dit c'est Marlène
sa sœur. Avouez que c'est confusant.
```

L'attribut *changetype* permet de modifier une entrée, il peut prendre les valeurs *modify*, *add*, *delete* ou encore *modrdn*. Pas défaut, s'il est omis, ce sera *add*.

Dans le cas de *modify* il faut choisir si l'attribut sera : ajouté, supprimé ou modifié.

- **add** – ajoute un attribut (si cet attribut existe déjà avec une autre valeur, une nouvelle paire attribut / valeur est ajoutée)

  ```
  add: <attribute-type>
  <attribute-type>: <value>
  ```

  Par exemple

  ```
  add: description
  description: My description
  ```

- **delete** – supprime tous les attributs de ce type sauf si l'on précise lequel on veut supprimer

  ```
  delete: <attribute-type>
  <attribute-type>: <value>
  ```

Par exemple

```
delete: work-phone

delete: mobileTelephonNumber
mobiletelephonnumber:
 +32 (0) 123 45 67 89
```

- **replace** – modifie l'attribut concerné. Si l'attribut a plusieurs valeurs, toutes les valeurs sont remplacées (s'il ne faut en remplacer qu'une, faire un delete suivi d'un add)

  ```
  replace: <attribute-type>
  <attribute-type>: <value>
  ```

  Par exemple

  ```
  replace: home-phone
  home-phone: +32 (2) 123 45 67
  ```

Par exemples :

```
dn:cn=Marlene Sassœur,ou=student,
 dc=example,dc=be
changetype : modify
add : telephonenumber
telephonenumber : 123 45 67 89

dn:cn=Marlene Sassœur,ou=student,
 dc=example,dc=be
changetype : delete
```

Modèle fonctionnel

Le modèle fonctionnel, est le protocole LDAP lui-même. Il fournit les moyens d'accéder aux données dans l'arbre LDAP. L'accès aux données consiste en l'authentification, les requêtes en lecture (recherche) et en écriture (ajout et mise à jour).

`OpenLDAP`

Implémentation du protocole : `OpenLDAP`

OpenLDAP est une implémentation libre du protocole LDAP.

L'installation se résume à :

```
# apt install slapd ldap-utils
```

Lors de l'installation, si aucun *rootDSE* n'est demandé, c'est que *debian* prend l'initiative d'utiliser *hostname.domainname*. Le plus simple si cela ne convient pas est de reconfigurer le paquet :

```
dpkg-reconfigure slapd
```

Le service se gère comme d'habitude avec `systemctl`.

Le répertoire `/etc/ldap` contient :

- les schémas dans `schemas` ;
- `ldap.conf` fichier de configuration éventuel pour les utilitaires `ldapfoo` ;
- un répertoire `slapd.d` contenant l'annuaire de configuration de *openLDAP*.

La configuration de l'annuaire est stockée… dans un autre annuaire nommé `cn=config`. Voir figure 5 page suivante.

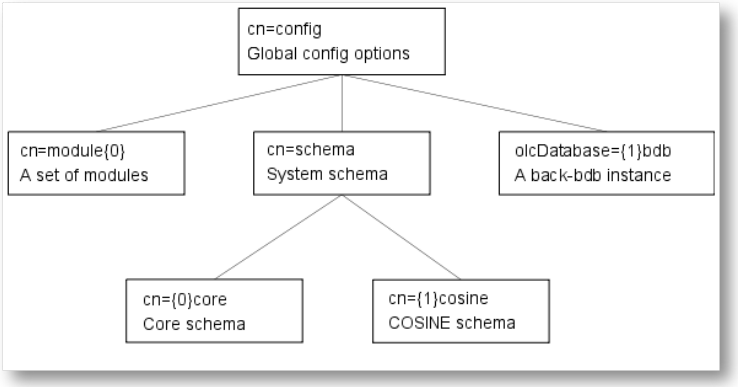


FIGURE 5 – Exemple d'un arbre de configuration

Schémas

Une des premières choses à faire est de vérifier que les schémas utiles sont bien pris en compte par l'annuaire. Ceci peut se faire en cherchant dans le nœud `cn=schema, cn=config`. Si ce n'est pas le cas, il suffit de les ajouter.

Les schémas disponibles se trouvent dans `/etc/ldap/schemas`. Notez bien que ce sont les schémas connus par *open ldap* pas les schémas pris en compte par l'annuaire.

Il peut arriver que les schémas disponibles ne soient pas suffisants pour répondre aux attentes de l'entreprise. Dans cette hypothèse, il faudra créer son propre schéma c'est-à-dire un ensemble de classes et d'attributs. Pour ce faire :

- choisir et structurer *objectId* et *attributeId* et les définir dans un fichier de schéma ;
- ajouter le schéma à l'annuaire.

La définition d'un objet peut avoir cette allure en utilisant le numéro d'entreprise de l'ESI.

```
#
# Local definition of yet another class
#
attributetype ( 1.3.6.1.4.1.23162
.504.2.1
  NAME 'yaName'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115
  .121.1.15{32768}
)

objectclass ( 1.3.6.1.4.1.23162
.504.1.1
  NAME ' yaObject '
  SUP top STRUCTURAL
  MUST ( cn $ sn )
  MAY ( yaName $ …)
)
```

ACL

Les ACL (*access control list*) OpenLDAP permettent de préciser **qui** a accès à **quoi** dans l'annuaire.

Le **qui** peut être choisi parmi : _*_, *self*, *anonymous*, *user* ou *regex*.

Specifier	Entities
*	All, including anonymous and authenticated users
anonymous	Anonymous (non-authenticated) users
users	Authenticated users
self	User associated with target entry
dn[.<basic-style>]=<regex>	Users matching a regular expression
dn.<scope-style>=<DN>	Users within scope of a DN

Les niveaux d'accès sont, quant-à eux choisis parmi : *none*, *disclose*, *auth*... et doivent être choisis dans l'ordre. Donner comme niveau d'accès *write*, implique de donner les accès : *read*, *search*, *compare*, *auth*, *disclose* et *none*.

Chaque niveau d'accès à une certaine **portée** (*scope*) qui peut-être : *base*, *one*, *subtree* ou *children*.

Level	Privileges	Description
none	=0	no access
disclose	=d	needed for information disclosure on error
auth	=dx	needed to authenticate (bind)
compare	=cdx	needed to compare
search	=scdx	needed to apply search filters
read	=rscdx	needed to read search results
write	=wrscdx	needed to modify/rename
manage	=mwrscdx	needed to manage

Le **quoi** détermine les entrées concernées par l'ACL. Il est composé de trois parties : une expression régulière déterminant le DN, un filtre de recherche et une liste de noms d'attributs séparés par des virgules. Les ACL peuvent être définies de manière dynamique, par le biais de fichier LDIF. C'est l'attribut `olcAccess` qui définit cet accès.

Exemples

Un exemple simple donnant l'accès à tous en lecture,

```
olcAccess: to * by * read
```

Une directive donnant la possibilité à l'utilisateur de la modifier, aux anonymes de s'authentifier, et aux autres de lire l'entrée.

```
olcAccess: to *
  by self write
  by anonymous auth
  by * read
```

Les entrées sous le sous-arbre `dc=com` sont accessibles en écriture, excepté pour les entrées sous `dc=example,dc=com` pour lesquelles un accès en lecture est autorisé.

```
olcAccess: to dn.children="dc=com"
  by * write
olcAccess: to dn.children=
 "dc=example,dc=com"
  by * read
```

Remarque

L'ordre dans lequel sont placées ces directives a de l'importance. Puisque cet ordre a de l'importance, la partie « quoi » est affublée automatiquement lors de l'ajout dans l'annuaire d'un numéro d'ordre placé entre accolades `{i}`.

Si l'on crée les entrées

```
olcAccess: to attrs=member,entry
  by dnattr=member selfwrite
olcAccess: to dn.children=
 "dc=example,dc=com"
  by * search
olcAccess: to dn.children=
```

```
 "dc=com"
  by * read
```

ce sont ces entrées qui sont enregistrées

```
olcAccess: {0}to attrs=member,entry
  by dnattr=member selfwrite
olcAccess: {1}to dn.children=
 "dc=example,dc=com"
  by * search
olcAccess: {2}to dn.children="dc=com"
  by * read
```

Recherche

La recherche la plus globale est la recherche « montre-moi tout ». Le filtre de recherche est alors `"(objectclass=*)"`

Un filtre LDAP a la forme suivant : `(attribut opérateur valeur)`.

L' attribut est le nom de l'attribut ! L' opérateur est choisi parmi :

- `=` pour l'égalité,
- `~=` pour les comparaisons approximatives,
- `<=` pour les comparaisons « inférieur ou égal »,
- `>=` pour les comparaisons « supérieur ou égal »

La partie valeur peut être une valeur absolue (par exemple `cn=juste`) ou une valeur reposant sur les wildcards (par exemple `cn=*blanc`).

Il est possible de regrouper des filtres élémentaires en utilisant les opérateurs booléens, `&` (ET), `|` (OU) et `!` (NON). Ces opérateurs utilisent la notation préfixée.

Pour rechercher les enregistrements ayant comme nom « foo » ou « bar », nous aurons

```
(|(cn=foo)(cn=bar))
```

Pour rechercher des enregistrements des unités organisationnelles « Sales » et « RH » qui ont dans leur *cn* « foo »

```
( &(|(ou=Sales)(ou=RH))(cn=*foo*) )
```

Il existe d'autres implémentations ; Microsoft Active Directory est une implémentation d'un annuaire qui « supporte » le protocole LDAP.

Le coin des commandes

`ldapsearch`

ldapsearch est l'outil de recherche dans l'annuaire. Il peut également rechercher dans la configuration de l'annuaire.

```
ldapsearch -LLL -Y EXTERNAL
  -H ldapi:/// -b "cn=config"
ldapsearch -LLL -Y EXTERNAL
 -H ldapi:/// -b "dc=example, dc=org"
```

- la première commande montre la configuration de l'annuaire, la seconde

montre l'arbre *example.org*
- cette commande ne demande pas de mot de passe et doit être exécutée par *root*

```
ldapsearch -LLL
  -D "cn=admin,dc=example,dc=org"
  -b "dc=example,dc=org"
  -x -W
```

- montre le contenu de l'arbre comme la commande précédente mais l'utilisateurice *admin* doit s'identifier.

```
ldapsearch -LLL -Y EXTERNAL
 -H ldapi:/// -b "cn=schema,cn=config"
```

- montre les schémas connus par l'annuaire.

```
ldapsearch -LLL
  -D "cn=user, dc=example,dc=org"
  -b "dc=example,dc=org"
  -x -W
  "(objectclass=*)
```

- fais une recherche de type « montre moi tout » à la base de l'annuaire.

Remarque : Afin d'éviter d'obtenir trop de résultats lors d'une recherche du genre `(objectclass=*)`, ldapsearch permet de définir des limites quant-à l'information retournée.

- `-l <value>`, définit en secondes, la durée d'attente maximale de la réponse à une demande de recherche.
- `-z <value>`, définit le nombre d'entrées maximal à récupérer lorsqu'une recherche aboutit.

Ces valeurs peuvent-être définie dans le fichier `ldap.conf` en utilisant les paramètres ; *timelimit* et *sizelimit*. Une valeur de 0 en ligne de commande annule les limites imposées dans le fichier de configuration des utilitaires ldap.

`ldapadd`

ldapadd permet l'ajout dans l'annuaire, que ce soit dans la partie configuration ou l'annuaire proprement dit.

Pour faire un ajout dans un annuaire, il faut au préalable créer un fichier LDIF contenant les enregistrements à ajouter.

```
$ cat add.ldif
dn: ou=rh,dc=example,dc=org
objectClass: organizationalUnit
ou: rh
```

L'inclusion dans l'annuaire se fait grâce à :

```
ldapadd -D "cn=admin,dc=example,dc=org"
  -W
  -f add.ldif
```

Pour ajouter des utilisateurices dans cet annuaire sous l'*OU rh*, un fichier LDIF pourrait être :

```
$ cat add.ldif
# Entrée Juste LEBLANC
dn: cn=Juste LEBLANC,ou=rh,
 dc=example,dc=org
objectclass : inetOrgPerson
cn: Juste LEBLANC
sn: Juste
mail: juste.leblanc@dev.null
mail: juste@leblanc.name
description: Il s'appelle Juste Leblanc.
 Ah bon,il a pas de prénom.
```

Serveur web

Installation et configuration **97**
Virtuals hosts **98**
ACL, *Access control list* **98**
Quand *HyperText Transfer Protocol* devient *Secured* **99**
Obtention d'un certificat 100
Le coin des commandes **102**

Un serveur web permet la mise à disposition de l'information sur un réseau

L'information mise à disposition par le serveur web est visible grâce à un navigateur (*browser*). C'est du texte balisé grâce au langage `HTML` (*HyperText Markup Language*). Le HTML permet de structurer la page : titres, sections, listes à puces… À ce langage HTML s'ajoute le `CSS` (*Cascading Style Sheet*) qui permettra de le mettre en forme : en rouge, en vert, dans une autre police, avec des marges plus grandes…

Les pages *web* telles qu'elles s'affichent dans nos navigateurs maintenant sont parfois réactives. Elles se sont plus simplement statiques. Elles réagissent aux clics de souris. Elles interagissent avec l'utilisateurice. Ceci parce que les navigateurs sont capables d'exécuter du code envoyé pas le serveur web. Ce langage est le `JS` (*Javascript*).

Le texte envoyé par un serveur web n'est pas simplement :

```
Hello world
```

mais pourrait-être[61] comme ci-dessous où il contient du HTML, du CSS et du Javascript[62] :

```
<!DOCTYPE html>
<style>
body {
    background-color: lightblue;
}
</style>
<html>
<title>Exemple</title>
```

[61] Exemple adapté de https://www.w3schools.com/

[62] Généralement toute l'information ne se trouve pas dans le même fichier et les styles (les fichiers CSS car ils peuvent être plusieurs) ainsi que le javascript éventuel sont inclus dans le fichier HTML… mais ce cours n'est pas un cours de développement web.

```
<body>

<p id="demo" onclick="myFunction()">
    Hello world
</p>

<script>
function myFunction() {
  var x = document
    .getElementById("demo");
  x.style.fontSize = "25px";
  x.style.color = "red";
}
</script>
</body>
</html>
```

Un serveur web est un service qui écoute, par défaut, sur les ports 80 et 443. Il est en attente d'une requête `HTTP` (*HyperText Transfert Protocol*) ou `HTTPS` (*HyperText Transfer Protocol Secured*) effectuée par le client (généralement le navigateur). Il existe différent type de requêtes HTTP|S, les plus connues étant `GET`, `HEAD` et `POST`.

Une requête HTTP demandant la page web à l'adresse `http://example.org` a l'allure suivante :

```
GET /index.html HTTP/1.1
Host: example.org
```

Une fois la requête effectuée, le serveur web cherche dans ses documents s'il a le fichier en question et le retourne.

Un serveur web peut générer les pages HTML/CSS/JS en utilisant un langage de script avant de les retourner. Par exemple `PHP`, *Ruby on rails*, `ASP`. Dans ce cas, les fichiers stockés sur le serveur sont écrits dans ces langages et sont interprètés avant d'être servis. Ceci demande des ressources au niveau du serveur.

Les serveurs web les plus répandus sont `Apache` et `nginx`.

Dans ces notes, nous traiterons uniquement de `Apache`.

Installation et configuration

Installation du paquet *apache2* pour le serveur web et de paquets spécifiques en fonction des modules que l'on veut rendre disponible.

Si apache doit comprendre le PHP, il est nécessaire d'installer *libapache2-mod-php7* par exemple.

Le fichier de configuration principal est `/etc/apache2/apache2.conf`.

On y trouve entre autres, la configuration de l'*essaim*. En effet, un serveur web a plusieurs processus qui écoutent et prennent en charge les demandes de connexion. Les paramètres correspondant sont :

```
KeepAlive on
MaxKeepAliveRequests 100
KeepAliveTimeout 10
Timeout 30
```

`KeepAlive` demande de garder la connexion ouverte un certain laps de temps pour éviter au client de réouvrir une connexion s'il a encore besoin de données. `MaxKeepAliveRequests` défini le nombre de requêtes concurrentes qui pourront être traitées. Les *time out* déterminent le temps (en secondes) avant qu'une connexion soit fermée.

Une autre information importante que l'on trouve dans le fichier de configuration est la possibilité de réécrire la configuration du serveur de manière spécifique pour un répertoire donné en y ajoutant un fichier `.htaccess`. Ceci est pratique lorsque la personne qui met en place le site web, n'a pas la possibilité d'accéder à la configuration de serveur web (chez un hébergeur par exemple)[63].

Le répertoire `mods-available` reprend tous les modules disponibles : ssl, proxy, php… Pour rendre un module disponible et utilisable par apache, utiliser la commande `a2enmod`. Cette commande a pour effet d'ajouter un lien *soft* du fichier `mods-available/<mod>.load` vers `mods-enabled/<mod>.load`.

Les répertoires `conf.d-available` et `conf.d-enabled` fonctionnent de la même manière pour des configuration particulière du serveur web.

[63] La recommandation d'apache est de n'utiliser cette fonctionnalité que lorsque l'on n'a pas accès au fichier de configuration principal. En effet, imposer la recherche et la lecture du fichier `.htaccess` pour **chaque** répertoire du serveur web est coûteux. Voir https://httpd.apache.org/docs/2.4/fr/howto/htaccess.html

Virtuals hosts

Un serveur web ne sert pas qu'un seul site web mais plusieurs. Pour chaque nom (derrière la même IP), il faut définir un hôte virtuel (*virtual host*). Par exemple, imaginons qu'à l'adresse IP 10.0.0.42 se trouvent les sites web

- *example.org*;
- *example.com* et ;
- *blog.example.org*.

Trois sites web différents sur un même serveur web.

Chacun des sites web devra avoir un fichier de configuration contenant au minimum les informations suivantes : le nom du site web et le répertoire associé dans lequel se trouvent les fichiers à servir.

La commande `a2ensite example.org.conf` rendra le site — le *virtual host* — disponible en créant un lien soft de `site-available` vers `sites-enabled`.

Par exemple :

```
# cat /etc/apache2/site-available/example.org.conf
<VirtualHost *:80>
    ServerName example.org
    ServerAlias www.example.org
    ServerAdmin webmaster@example.org
    DocumentRoot /var/www/html/example.org

    ErrorLog ${APACHE_LOG_DIR}/example.org-error.log
    CustomLog ${APACHE_LOG_DIR}/example.org-access.log combined
</VirtualHost>
```

TODO: *reverse proxy*

ACL, *Access control list*

Le **contrôle d'accès** fait référence à tout concept d'accès à une ressource quelconque. Il est distinct du processus d'**authentification et d'autorisation**[64].

Les **ACL** (*access control list*) définissent

[64] Extrait de la documentation officielle d'apache https://httpd.apache.org/docs/2.4/fr/howto/access.html

des restrictions d'accès à une ressources. Ces ACL sont régulièrement écrites dans des fichiers `.htaccess` dès lors que l'on n'a pas accès à la configuration du serveur. Sinon, elles peuvent directement être écrites dans le fichier de configuration du *virtual host*.

Dans la littérature, elles s'illustrent souvent comme ci-dessous pour empêcher l'accès à une ressource (typiquement un fichier ou un répertoire) en fonction d'une IP ou d'un nom d'hôte. Ici la ressource n'est accessible que localement :

```
Order deny, allow
Deny from all
Allow from 127.0.0.1/8
```

Ceci fait référence au module *mod_access_compat* dont les directives sont **devenues obsolètes** depuis le refonte *d'authz* (voir le module *mod_authz_host* cfr.(Internet, s.d.-a)) pour la version 2.4 d'Apache.

La directive précédente se réécrira dans sa nouvelle version (voir documentation Apache[65]) en utilisant le module *mod_authz_host* :

```
Require ip 127.0.0.1
```

La directive **Require** s'utilise comme suit :

```
Require [not] host address
Require [not] ip IP
Require local
```

La dernière, `Require local`, autorise l'accès si l'IP est `127.0.0.1/8` **ou** `::1` **ou** si les IP du client et du serveur sont identiques. Bref, si c'est en local.

Cette directive se trouve généralement à l'intérieur d'une section `<Directory>`, `<Files>` ou `<Location>`. Les directives se trouvant à l'intérieur d'une telle section ne concerne que la section.

`Directory` pour le répertoire renseigné.
`Files` pour les fichiers renseignés.
`Location` pour les *urls* renseignées.

Quand *HyperText Transfer Protocol* devient *Secured*

En `HTTP` toutes les requêtes transitent en clair et sont visibles. Depuis 2017 toute page web proposant l'envoi d'un formulaire de saisie de mot de passe ou de numéro de carte de crédit est renseigné comme **non sécurisé**.

`HTTPS` est une connexion `HTTP` encapsulée dans `TLS` (*Transport Layer Security*). La sécurisation de la connexion permet d'authentifier le serveur, de garantir la confidentialité et l'intégrité des données échangées. Le contenu est chiffré ainsi que la requête, les *hearders* (entêtes), *cookies*.

Le port utilisé par défaut est le port **443**.

En `HTTPS`, dès que le client a résolu le nom et trouvé l'IP correspondante, il se connecte au port 443 et présente la liste des protocoles qu'il connait pour la sig-

[65] https://httpd.apache.org/docs/2.4/fr/upgrading.html

nature, l'échange de clés et le chiffrement des données. Le serveur va alors retourner un **certificat** contenant une clé publique, le mécanisme d'échange de clés et quel algorithme sera utilisé pour le chiffrement. Ce certificat est vérifié par le client en contactant l'autorité de certification (CA). L'autorité de certification a signé le certificat avec sa clé privée et la transmis au site web. Le client peut donc le présenter à l'autorité de certification afin de savoir si le serveur est bien qui il prétend être. Le certificat me garantit que le site auquel je me connecte n'est pas usurpé.

C'est seulement alors que le canal de communication chiffré (avec une clé de chiffrement symétrique pour la session) est étable et que la communication peut avoir lieu.

> Aie confiance. Crois-en moi.

Pour avoir confiance en une connexion, il faut :

- avoir confiance dans navigateur ;
- avoir confiance dans l'autorité de certification (**CA** *certification authority*) ;
- avoir confiance dans le protocole TLS.

La confiance dans le navigateur est assez personnelle et ne devrait pas n'être qu'une préférence. Fait-on confiance en Mozilla Firefox, Google Chrome, Microsoft Edge…

Pour l'autorité de certification, elles sont connues des navigateurs et bien que certaines ait été compromises, on peut le faire confiance.

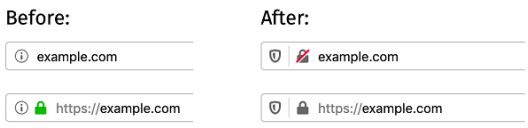
Lorsque l'on fait une demande de certificat, l'autorité vérifie que le demandeur ou la demandeuse est bien propriétaire du site. Les vérifications de base peuvent se faire via :

http A-t-on la main sur l'hébergement web ?
Vérification *via* le dépôt d'un fichier fourni pas le CA.

dns A-t-on la main sur la zone DNS ?
Vérification *via* le dépôt d'un enregistrement particulier dans la zone fourni par le CA.

mail A-t-on la main sur les mails ?
Vérification par la création d'une adresse mail particulière également fournie par le CA.

Au delà de ces vérifications, il est possible d'avoir un certificat *EV SSL, extended validation* pour lesquels l'autorité de certification procédera à un contrôle d'identité complet et normalisé ; droits exclusifs d'utilisation du domaine, existence légale, opérationnelle et physique et prouver que l'autorité a autorisé l'émission du certificat. Depuis 2019, les sites possédant un tel certificat ne se distingue plus des autres ce qui était le cas avant par l'affichage d'une « barre verte ».



Source[66]

Obtention d'un certificat

L'obtention d'un certificat se fait *via* une autorité de certification ou *via* **Let's En-**

[66] http://www.ullm.org/mozilla-firefox-70-affichera-de-nouveaux-indicateurs-de-securite/

crypt.

Autorité de certification

La procédure est semblable quelle que soit l'autorité de certification et se résume à :

- créer une `CSR` (*Certificate Signing Request*), un bloc de texte chiffré qui précise qui l'on est et quel est le nom de domaine concerné.

```
openssl req -nodes \
    -newkey rsa:2048 \
    -sha256 -keyout my.key \
    -out my.csr
```

- choisir la méthode de vérification http, dns ou mail pour les certificats standard. Pour les autres, il faudra fournir en plus certains documents.
- récupérer le certificat et le déposer sur l'hébergement.

Let's Encrypt

Let's Encrypt est une autorité de certification sans but lucratif qui délivre également des certificats. Outre qu'ils soient gratuit, Let's Encrypt fourni des scripts aidant à la génération des certificats et à leur renouvellement. Il s'agit de script utilisant le protocole `ACME` (*Automatic certificate management environment*). Ce protocole permet l'automatisation du renouvellement et de la demande de certificat.

Let's Encrypt recommande `Certbot`[67] comme client ACME. *dehydrated*[68] est également un client très pratique d'usage.

Un fois le certificat obtenu et déposé sur le serveur, le *virtual host* peut être adapté pour prendre en charge *https* en ajoutant la section :

```
# cat /etc/apache2/site-available/example.org.conf
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName example.org
    ServerAlias www.example.org
    ServerAdmin webmaster@example.org
    DocumentRoot /var/www/html/example.org

    SSLEngine on
    SSLCertificateFile /elsewhere/yourcert.pem
    SSLCertificateKeyFile /elsewhere/yourcert.key

    ErrorLog ${APACHE_LOG_DIR}/example.org-error.log
    CustomLog ${APACHE_LOG_DIR}/example.org-access.log combined
</VirtualHost>
</IfModule>
```

[67] https://certbot.eff.org/
[68] https://dehydrated.io/

Utilisation de *dehydrated* pour gérer ses certificats TODO

Le coin des commandes

`nc`, `ncat` ou `telnet`

Ces commandes établissent une connexion TCP avec un serveur. Il est par exemple possible de faire une requête HTTP/GET avec *netcat* (`nc`)

```
printf 'GET / HTTP/1.1\r\n\
    Host:esigoto.info\r\n\r\n'
 | nc esigoto.info 80
```

et en HTTPS,

```
printf 'GET / HTTP/1.1\r\n\
Host:esigoto.info\r\n\r\n'
 | ncat --ssl esigoto.info 443
```

`a2ensite` et `a2dissite`

a2ensite rend un site — *virtual host* — disponible en créant un lien *soft* de fichier de configuration dans `/etc/apache2/sites-availables` vers `/etc/apache2/sites-enabled`. Il est bien sûr nécessaire de demander à *apache* de relire sa configuration.

a2dissite exécute l'action invers.

```
a2ensite example.org.conf
a2dissite example.org.conf
```

`a2enmod` et `a2dismod`

a2enmod et *a2enmod* fonctionne de la même manière pour les modules.

```
a2enmod ssl
a2dismod ssl
```

Tip retirer les commentaires dans un fichier de configuration.

```
grep -v "^[#|$]" <fichier>
  | grep .
```

```
grep -v "^[#|$]" \
  /etc/apache2/apache2.conf
  | grep .
```

Index

.htaccess, 97, 98

ACL, 92
Active Directory, 73
adduser, 9
AES, 45
Apache, 95
apt, 20, 21
ARP, 27, 31
arp, 30

bash, 33
bat, 50
bind, 64
blowfish, 45

CA, 100
cat, 50
cd, 15
chacha20, 45
chage, 10
chgrp, 15
chmod, 15
cloudfare, 71
CSR, 100
css, 95
Ctrl-Z, 18

deluser, 9
df, 15
diffie-hellman, 44
dig, 32, 69
DN, 88
DNS, 32, 60
dns, 30
DNSSEC, 60, 64, 69
doh, 59, 71
domaine, 76
dot, 59
DSA, 44
du, 15

ecdsa, 44
ed25519, 44
editor, 3
esperluette, 17
Ethernet, 27

FAI, 59
filesystem, 11
fqdn, 30

Grafana, 50
grep, 43
group, 4
grub2, 21

head, 43
hostname, 30, 31
hosts, 51
html, 95
htop, 18
HTTP, 96

ICMP, 27
ifconfig, 30
InfluxDB, 50
IP, 27
IPv4, 27
IPv6, 27, 28

Javascript, 95

Kerberos, 73, 74
kill, 17
knock, 46
knockd, 46
knocking, 46
KSK, 60

LAN Manager, 73
last, 49, 50
LDAP, 74, 86
ldap, 81
ldapsearch, 93
LDIF, 89
Let's Encrypt, 100
log, 49
ls, 14

MAC, 29
man, 2
miroir, 20
mkfifo, 15
modèle TCP-IP, 25
mount, 15

nc, 102
NetBIOS, 72, 74
netcat, 102
netstat, 31
nginx, 95
NIS, 51
NSEC, 60
NSEC3, 60
nsswitch, 53, 77

openssl, 47

PAM, 10, 77, 83
passphrase, 47
passwd, 4, 8
pidof, 19
ps, 17
pstree, 18

resolv.conf, 32, 66
resolver, 54
rndc, 64
root, 3
routage, 29
route, 30
RSA, 44

Samba, 73
shadow, 4
shell, 33
SIGHUP, 18
signal, 78
signaux, 17
SMB, 73

source.list, 20
ssh, 44, 48
ssh-add, 47
ssh-agent, 47
ssh-copy-id, 47
ssh-keygen, 47
state, 16
su, 10
sudo, 7, 10
systemctl, 22
tail, 43
TCP, 25
telegraf, 50
top, 18
touche, 15
umount, 15
UPD, 25
user, 4
useradd, 9
userdel, 9
usergroup, 9
who, 50
winbind, 76
workgroup, 76
yellow pages, 51
ZSK, 60, 71

Bibliographie

Bettens, P. (s.d.). *Mise en place d'un DNS menteur*. Récupéré le 11 mars 2021 de https://blog.namok.be/?post/2017/03/05/mise-en-place-dns-menteur

Bettens, P. (2021, 11 mars). *Quand ton serveur DNS te bloque ou te ment*. https://blog.namok.be/?post/2016/10/18/quand-ton-serveur-dns-te-bloque-ou-te-ment

Bortzmeyer, S. (s.d.). *RFC 9156: DNS Query Name Minimisation to Improve Privacy*. Récupéré le 10 janvier 2022 de https://www.bortzmeyer.org/9156.html

Codutti, M. (2003). *Administration Système, ULB INFO151*.

Dawson, T., Purdy, G. et BAUTTS, T. (Janvier 2005). *Administration réseaux sous Linux* (3 éd.). O'Reilly.

Domain Name System (DNS) Parameters. (s.d.). Récupéré le 5 mars 2021 de http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

Eckstein, R., Collier-Brown, D. et Kelly, P. (2000). *Samba. installation et mise en œuvre*. O'REILLY.

Goffinet, F. (s.d.). *Introduction aux adresses IPv6*. https://cisco.goffinet.org/ccna/ipv6/introduction-adresses-ipv6/

Hunt, G. (Janvier 2000). *TCP/IP Administration des réseaux* (3 éd.). O'REILLY.

iana (internet assigned numbers authority), the global coordination of the DNS Root, IP addressing…. (s.d.). Récupéré le 21 février 2021 de https://www.iana.org/

Internet. (s.d.-a). *Apache HTTP Server Versions 2.4 Documentation*. https://httpd.apache.org/docs/2.4/fr/

Internet. (s.d.-b). *Filesystem Hierarchy Standard (pdf)*. Récupéré le 12 février 2021 de https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf

Internet. (s.d.-c). *Licence creative Common BY-NC-SA 4.0*. Récupéré le 12 février 2021 de http://creativecommons.org/licences/by-nc-sa/4.0/

Internet Protocal Version 6 Address Space. (s.d.). Récupéré le 21 février 2021 de https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml

Labs, Nl. (s.d.). *Unbound by NLnet Labs*. Récupéré le 12 février 2024 de https://unbound.docs.nlnetlabs.nl/en/latest/

Root zone database. (s.d.). Récupéré le 4 mars 2021 de https://www.iana.org/domains/root/db

Wikipedia. (s.d.). *Filesystem Hierarchy Standard*. Récupéré le 12 février 2021 de https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard