

Chapitre 7

Le réseau

1. TCP/IP

1.1 Bases

L'origine de **TCP/IP** provient des recherches du **DARPA** (*Defense Advanced Research Project Agency*) qui débutent en 1970 et débouchent sur **ARPANET**. Dans les faits, le DARPA a financé l'université de Berkeley qui a intégré les protocoles de base de TCP/IP au sein de son système **UNIX BSD 4**.

TCP/IP s'est popularisé grâce à son interface générique de programmation d'échanges de données entre les machines d'un réseau, les primitives **sockets**, et l'intégration de protocoles applicatifs. Les protocoles de TCP/IP sont supervisés par l'**IAB** (*Internet Activities Board*) lui-même supervisant deux autres organismes :

- L'**IRTF** (*Internet Research Task Force*) qui est responsable du développement des protocoles.
- L'**IETF** (*Internet Engineering Task Force*) qui est responsable du réseau Internet.

Les adresses réseau sont distribuées par le **NIC** (*Network Information Center*) et en France l'**INRIA**. L'ensemble des protocoles de TCP/IP est décrit dans les documents **RFC** (*Request For Comments*) (voir le RFC 793).

- La couche inférieure est **IP** (*Internet Protocol*).
- La couche de transport est **TCP** (*Transmission Control Protocol*) ou **UDP** (*User Datagram Protocol*).
- Les couches supérieures sont les couches des protocoles applicatifs, par exemple :
 - **NFS** (*Network File System*) : partage de fichiers à distance.
 - **DNS** (*Domain Name System*) : association hôte<->IP.
 - **FTP** (*File Transfer Protocol*) : transfert de fichiers.
 - **TELNET** : émulation d'un terminal de type texte...

La version du protocole IP représenté est la v4. Le futur, déjà présent, est le protocole **IPv6**. Compatible IPv4, il propose un adressage sur 128 bits (16 octets)

permettant d'étendre les capacités du réseau notamment en matière de taille et d'adressage.

1.2 Adressage

1.2.1 Classes

Il est important de savoir avant l'installation dans quel type de réseau doit s'intégrer le nouveau serveur, TCP/IP bien sûr, mais il faut déjà lui réservier une adresse IP, un hostname (nom de machine réseau), connaître les diverses passerelles, le nom de domaine, la classe utilisée et le masque de sous-réseau netmask.

Voici un bref rappel sur les classes IP. Une adresse IP est définie sur 32 bits et représentée par quatre nombres séparés par des points : **n1.n2.n3.n4**. Cette adresse est constituée de deux parties qui définissent l'adresse réseau et l'hôte dans le réseau.

Distinguez, suivant les cas, quatre ou cinq classes d'adresses : A, B, C, D et E, mais seules les trois premières nous intéressent.

Légende : N et h sont des bits, N identifiant du réseau h identifiant de la machine.

Classe A : ONNNNNNN hhhhhh hh hh hh hh soit 1.x.x.x à 127.x.x.x.
n1 est compris entre 1 et 127.
16777214 hôtes, 127 réseaux.

Classe B : 10NNNNNN NNNNNNNN hhhhhh hh hh hh hh soit de 128.0.x.x à 191.255.x.x.
n1 est compris entre 128 et 191.
65534 hôtes, 16382 réseaux.

Classe C : 110NNNNN NNNNNNNN NNNNNNNN hh hh hh hh soit de 192.0.0.x à 223.255.255.x.
n1 est compris entre 192 et 223.
254 hôtes, 2097150 réseaux.

Classe D : Commence par 1110, pour la multidiffusion IP de 224 à 239.
Classe E : Commence par 1111, pour expérimentation IP de 240 à 255.

Il existe des adresses d'hôtes qui ne peuvent pas être exploitées. Par exemple dans la classe C on ne peut avoir que 254 hôtes, alors que l'identifiant de la machine est codé sur 8 bits (donc 256 valeurs). C'est que l'adresse 0 représente l'adresse du réseau, et l'adresse 255 celle du **broadcast** (multidiffusion).

Notez que les adresses suivantes ne doivent pas être routées sur Internet et sont réservées aux réseaux locaux.

- 10.0.0.0 - 10.255.255.255 (10/8)

- 172.16.0.0 - 172.31.255.255 (172.16/12)
- 192.168.0.0 - 192.168.255.255 (192.168/16)

L'adresse 127.0.0.1 est l'adresse de loopback ou bouclage : elle représente la machine elle-même, ainsi que le sous-réseau 127.0.0.0/8.

1.2.2 Sous-réseaux

De plus, il est possible de découper ces réseaux en sous-réseaux à l'aide de masques permettant un découpage plus fin des adresses. Un **netmask** est un masque binaire qui permet de séparer immédiatement l'adresse du réseau et du sous-réseau de l'adresse de l'hôte dans l'adresse IP globale. Les masques prédéfinis sont :

- **Classe A** : 255.0.0.0
- **Classe B** : 255.255.0.0
- **Classe C** : 255.255.255.0

Pour communiquer directement entre eux les hôtes doivent appartenir à un même réseau ou sous-réseau. Calculer un sous-réseau est assez simple. Voici un exemple pour un réseau de classe C.

- Réseau : 192.168.1.0
- Adresse de réseau : 192.168.1.255
- Masque de réseau : 255.255.255.0

Calculer un masque de sous-réseau :

- Pour calculer le masque de sous-réseau, vous devez tout d'abord déterminer combien de machines vous souhaitez intégrer dans celui-ci. Un réseau de classe C permet d'intégrer 254 machines (0 et 255 étant réservés). Vous souhaitez créer des réseaux contenant 60 machines. Ajoutez 2 à cette valeur pour les adresses réservées (adresse du sous-réseau et adresse de broadcast) ce qui donne **62**.
- Une fois le nombre de machines déterminé, trouvez la puissance de deux exacte ou juste supérieure au nombre trouvé. 2 à la puissance 6 donne **64**.
- Écrivez le masque en binaire, placez tous les bits du masque de réseau de classe C à 1 et placez à 0 les 6 premiers bits du masque correspondant à la partie machine : **11111111 11111111 11111111 11000000**
- Convertissez ce masque en décimal : **255.255.255.192**, et calculez l'ensemble des sous-réseaux possibles. Comme vous êtes dans un réseau de classe C, vous pouvez encore faire varier les deux derniers bits de la partie machine :
 - 00xxxxxx : 255.255.255.0
 - 01xxxxxx : 255.255.255.64
 - 10xxxxxx : 255.255.255.128
 - 11xxxxxx : 255.255.255.192

- Au final, vous obtenez quatre sous-réseaux de 62 machines, soit 248 machines. Vous tombez bien sur 256 si vous rajoutez les quatre adresses de broadcast et les quatre adresses de réseau.

1.2.3 Routage

Le masque de réseau permet de déterminer si une machine destinataire est sur le même réseau que vous ou non. Il faut indiquer le chemin que doivent prendre les paquets IP pour rejoindre leur destination. Si votre machine est un poste client disposant d'une seule carte réseau et que ce réseau ne comporte qu'un seul routeur (cas classique d'une connexion vers Internet) alors vous devez créer deux routes. La première est celle indiquant quelle carte réseau doivent emprunter les paquets pour accéder au reste du réseau (au sous-réseau), la seconde quelle route doivent emprunter les paquets pour sortir du réseau. Généralement, on parle de route par défaut quand un seul routeur est présent.

- Vers réseau1 -> utiliser interface réseau gauche.
- Vers réseau2 -> utiliser interface réseau droite.
- Vers autres -> utiliser interface réseau droite vers routeur1.

Exemple : Réseau1 de classe C 192.168.1.0 sur eth0, Réseau2 de classe B 172.16.0.0 sur eth1, adresse de routeur 192.168.1.254.

Réseau	Masque	Interface	Passerelle
192.168.1.0	255.255.255.0	eth0	eth0
172.16.0.0	255.255.0.0	eth1	eth1
0.0.0.0	0.0.0.0	eth0	192.168.1.254

Tous les paquets réseaux vers 192.168.1.0 transiteront par eth0. Tous les paquets à destination de 172.16.0.0 transiteront par eth1. Par défaut, tous les autres paquets pour les réseaux non spécifiés transiteront par eth0 et seront traités par la passerelle 192.168.1.254 qui routera les paquets.

1.2.4 IPv6

Saturation d'IPv4

Les dernières adresses IPv4 publiques sont arrivées à saturation le 3 février 2011 alors qu'elles ne sont pas toutes utilisées, du fait du mode de répartition par classes et sous-réseaux. Par exemple, les seuls sous-réseaux de classe A représentent près de deux milliards d'adresses IP, probablement pas toutes utilisées ou aux mains d'organismes qui ne les utilisent pas toutes. Il en est de même pour les sous-réseaux de classe B pour un milliard d'adresses.

Un grand nombre d'IP sont réservées à des usages précis, comme par exemple les IP multicast, et ne peuvent pas être affectées à un ordinateur.

Une mauvaise gestion initiale, la multiplication des terminaux, notamment les terminaux mobiles divers, et la multiplication des connexions Internet des particuliers ont fait exploser la demande, ce qui a conduit à la saturation.

Diverses méthodes ont été appliquées pour tenter de repousser la date de saturation, notamment en faisant disparaître la notion de classes sur les IP publiques, ou l'utilisation de NAT permettant, via une translation d'adresse, de proposer des services accessibles au public sur des machines présentes dans des sous-réseaux privés. C'est ainsi que vous pouvez installer des serveurs web ou autres sur vos propres PC à la maison, qui ont une IP dans votre sous-réseau privé, mais accessibles depuis l'extérieur via l'IP publique et les fonctions NAT de votre boîtier Internet.

Les adresses IPv4 inutilisées encore aux mains de leurs propriétaires originaux font maintenant l'objet d'un commerce lucratif.

Ceci ne signifie pas dans l'immédiat la disparition d'IPv4 : ces adresses IP resteront bien entendu très largement utilisées pendant quelques années, le temps que toutes les IP inutilisées ne le soient plus, ce qui devrait se produire entre 2011 et 2015. Les adresses IPv4 publiques sont amenées à disparaître ou à cohabiter avec autre chose.

Adressage IPv6

Les adresses IPv6 vont remplacer les adresses IPv4. Elles sont cette fois codées sur 128 bits et leur nombre est tel que chaque être humain pourrait en obtenir des milliards de milliards sans arriver à saturation. Ainsi le NAT devient inutile et le routage devient bien plus simple.

L'adresse est décomposée en 8 groupes de deux octets écrits en hexadécimal et séparés par des « : », soit 39 caractères :

■ 2001:0e36:2ed9:d4f0:021b:0000:0000:f81d

Un à trois zéros non significatifs peuvent être omis :

■ 2001:e36:2ed9:d4f0:21b:0:0:f81d

Un ou plusieurs blocs de 16 bits (2 octets) nuls peuvent être omis tout en conservant les « : » de chaque côté. Une seule substitution est possible :

■ 2001:e36:2ed9:d4f0:21b::f81d

Le préfixe par défaut chez Free est /64. Ceci signifie que votre fournisseur d'accès, par exemple Free en France, va vous fournir une adresse sous la forme 2a01:e36:2ed9:d4f0::/64, soit, de manière développée :

```
# 2a01:e36:2ed9:d4f0:0000:0000:0000:0000/64
```

Ce sous-réseau est le vôtre et est automatiquement routé par la Freebox. Il ne peut pas (encore) être divisé en d'autres sous-réseaux. Vous pouvez créer autant d'adresses IPv6 que vous le souhaitez dans ce sous-réseau, soit environ 264 hôtes : 18 446 744 073 709 551 616 adresses IP. Ces IP seront directement accessibles depuis l'extérieur, sans NAT, mais bien entendu la protection par firewall est possible.

Utilisation sous Linux

Ce livre se base sur la configuration des interfaces et des services pour une utilisation en IPv4. Cependant le principe est exactement le même en IPv6. Linux est entièrement compatible IPv6, ceci nativement. Les commandes ifconfig, route, ping, etc. acceptent IPv6, ainsi que les fichiers de configuration des interfaces propres à chaque distribution. La suite fournit quelques exemples adaptés.

1.3 Configuration

1.3.1 Cas des distributions de type Red Hat/Fedora

Red Hat propose des outils pour configurer le réseau de base sans passer par la manipulation des fichiers de configuration. Le programme **netconfig** est en mode dynamique, routeur, nom d'hôte, serveur de noms).

```
# netconfig --device eth0
```

La commande graphique **system-config-network** lance une interface plus complète.

Les distributions Mandriva et openSUSE reprennent le même principe que ce qui est exposé ci-après, mais la syntaxe et la position des fichiers peuvent varier. Vous vous reporterez à la documentation de votre distribution pour plus de détails.

Interfaces réseau

La configuration de base d'une interface réseau se fait à l'aide de la commande **ifconfig**. Cependant la plupart des distributions utilisent des scripts d'administration et des fichiers de configuration qui simplifient énormément les choses car ils configurent à la fois l'interface réseau et les routes.

Configuration de eth0 pour l'adresse de classe C 192.168.1.2

```
| ifconfig eth0 inet 192.168.1.2 netmask 255.255.255.0  
| ifconfig eth0 192.168.1.2
```

Activation de l'interface réseau eth0 :

```
| ifconfig eth0 up
```

Arrêt de l'interface réseau eth0 :

```
| ifconfig eth0 down
```

Affichage des informations de eth0 :

```
| # ifconfig eth0  
eth0      Lien encap:Ethernet HWaddr 00:XX:XX:XX:XX:XX  
          inet adr:192.168.1.60 Bcast:192.168.1.255 Masque:255.255.255.0  
          adr inet6: fe80::21b:fcff:fec9:f81d/64 Scope:Lien  
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
          RX packets:16522 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:13631 errors:0 dropped:0 overruns:0 carrier:2  
          collisions:0 lg file transmission:1000  
          RX bytes:17732221 (16.9 Mb) TX bytes:1648879 (1.5 Mb)
```

Notez que l'adresse IPv6 apparaît dans le résultat sur la ligne inet6.

Configuration d'une adresse IPv6 :

```
| # ifconfig eth0 inet6 add fe80::21b:fcff:fec9:f81d/64
```

Affichage de toutes les interfaces réseaux activées :

```
| ifconfig
```

Affichage de toutes les interfaces réseaux activées ou non :

```
| ifconfig -a
```

Le mieux reste d'utiliser les scripts **ifup** et **ifdown**. Ceux-ci se basent sur les fichiers présents dans /etc/sysconfig/network-scripts/. Ces fichiers de configuration d'interface se nomment ifcfg-xxx où xxx est le nom de l'interface réseau, comme eth0 :

```
| DEVICE=eth0  
| IPADDR=192.168.1.2  
| NETMASK=255.255.255.0  
| NETWORK=192.168.1.0  
| BROADCAST=192.168.1.255
```

```
ONBOOT=yes
BOOTPROTO=static
```

Les paramètres parlent d'eux-mêmes. Les valeurs **NETWORK** et **BROADCAST** sont optionnelles si **IPADDR** et **NETMASK** sont renseignés (dans ce cas le calcul est automatique) ou si **DHCP** est utilisé. **BOOTPROTO** indique comment monter l'interface, soit **static**, soit **dhcp**. La valeur **bootp** peut aussi être utilisée. Dans le cas de DHCP, le fichier peut ressembler à ceci :

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Dans le cas d'une configuration statique, **IPADDR** et **NETMASK** sont obligatoires :

```
DEVICE=eth0
IPADDR=192.168.1.2
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=static
```

ONBOOT détermine si l'interface doit être automatiquement activée au démarrage de la machine.

Pour IPv6, vous devez utiliser les variables suivantes :

```
IPV6INIT=yes
IPV6ADDR=<IPv6-IP-Address>
IPV6_DEFAULTGW=<IPv6-IP-Gateway-Address>
```

IPV6INIT indique si IPv6 est activé ou non sur cette interface. Les deux variables suivantes indiquent l'adresse IP et sa passerelle.

Une fois le fichier correctement renseigné, on utilise les commandes **ifup/ifdown** :

Activation de l'interface eth0 :

```
ifup eth0
```

Arrêt de l'interface eth0 :

```
ifdown eth0
```

Paramètres généraux

Le fichier **/etc/sysconfig/network** contient les paramètres généraux du réseau.

```
NETWORKING=yes
HOSTNAME=poste1.monreseau.org # nom complet
GATEWAY=0.0.0.0 # passerelle par défaut
NISDOMAIN= # nom du domaine NIS
NETWORKING_IPV6=yes
```

- **NETWORKING** : activation ou non du réseau.
- **HOSTNAME** : nom de domaine complet FQDN.
- **GATEWAY** : adresse IP de la passerelle.
- **GATEWAYDEV** : interface réseau permettant d'accéder à la passerelle.
- **NISDOMAIN** : cas d'un domaine NIS.
- **NETWORKING_IPV6** : activation ou non du support d'IPv6.

1.3.2 Machines de type Debian

Le fichier de configuration des interfaces réseaux sous Debian (et Ubuntu) est situé dans /etc/network/interfaces. Il n'a pas du tout le même format que précédemment :

```
# cat interfaces
auto lo eth0 eth1
iface lo inet loopback

iface eth0 inet static
    address 192.161.1.60
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.1

iface eth1 inet dhcp
```

Cet exemple montre trois types d'interfaces :

- l'interface lo de loopback.
- l'interface eth1 en dhcp, ne nécessitant pas de configuration plus avancée.
- l'interface eth0 configurée statiquement.

La syntaxe générale d'une déclaration est la suivante :

```
interface nom type mode
```

Avec une configuration statique, précisez les différents paramètres avec les mots clés suivants :

- **address** : l'adresse IP.
- **netmask** : le masque de sous-réseau.
- **broadcast** : l'adresse de broadcast.

- **gateway** : la passerelle par défaut.

La ligne **auto** précise les interfaces qui seront automatiquement activées au démarrage.

Pour IPv6 la configuration est identique : il suffit simplement de remplacer **inet** par **inet6**.

Le fichier **/etc/hostname** contient le nom de la machine :

```
# cat /etc/hostname
slyserver
```

1.3.3 Routage

Avec l'utilisation des fichiers et des commandes précédentes, il n'y a pas besoin de créer un routage spécifique car la passerelle par défaut est déjà présente (cf. paramètres généraux) et les routes pour les interfaces réseaux sont automatiquement mises en place par **ifup**. Cependant on peut utiliser la commande **route**.

Affiche les routes actuelles :

```
route
netstat -nr
```

Dans l'exemple suivant, les interfaces **vmnet1** et **vmnet8** sont des interfaces virtuelles issues de la configuration réseau de VMWare.

```
# netstat -rn
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic   MSS Fenêtre irtt
Iface
192.168.211.0  0.0.0.0        255.255.255.0    U        0 0          0
vmnet8
192.168.1.0    0.0.0.0        255.255.255.0    U        0 0          0
eth0
172.16.248.0   0.0.0.0        255.255.255.0    U        0 0          0
vmnet1
169.254.0.0    0.0.0.0        255.255.0.0     U        0 0          0
eth0
127.0.0.0      0.0.0.0        255.0.0.0       U        0 0          0
lo
0.0.0.0         192.168.1.1    0.0.0.0        UG       0 0          0
eth0
```

Ajout de l'entrée loopback :

```
route add -net 127.0.0.0
```

Ajoute la route vers le réseau 192.168.1.0 passant par eth0.

Netmask peut être omis.

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Ajoute la passerelle par défaut vers le routeur :

```
route add default gw 192.168.1.254
```

Supprime la route vers le réseau 172.16.0.0 :

```
route del -net 172.16.0.0 eth0
```

La gestion des routes IPv6 est identique, mais il faut cependant indiquer le type d'adresse utilisé à la commande avec l'option -A :

```
route -A inet6 add <ipv6> gw <ipv6>
```

1.4 Outils réseaux

1.4.1 FTP

Il est utile de connaître la commande **ftp**, associée au protocole de même nom (*File Transfer Protocol*). Elle permet le transfert de fichiers entre deux machines. Elle prend comme paramètre le nom de la machine distante. Pour que la commande **ftp** fonctionne, il faut que le service **ftp** fonctionne sur la machine distante et sur le port 21.

Voici un exemple (peu pratique) de connexion avec erreur et nouvel essai.

```
ftp> open  
(to) machine  
Connected to machine.  
220 machine FTP server (Digital UNIX Version 5.60) ready.  
Name (machine:root): root  
331 Password required for root.  
Password:  
530 Login incorrect.  
Login failed.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> user  
(username) root  
331 Password required for root.  
Password:  
230 User root logged in.  
ftp> pwd  
257 "/" is current directory.
```

Le plus simple est tout de même :

```
$ ftp machine
Connected to machine.
220 machine FTP server (Digital UNIX Version 5.60) ready.
Name (machine:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Voici une liste de commandes ftp.

Commande	Action
open	Suivi d'un nom de machine, ouvre une connexion sur la machine spécifiée.
user	Saisie de l'utilisateur distant pour une connexion.
quit	Fin de la connexion et fin de la commande ftp.
ascii	Transfert des fichiers en mode ASCII (conversion des caractères spéciaux et fin de ligne en MS et Unix par exemple).
binary	Transfert des fichiers en mode binaire.
glob	Supprime l'interprétation des caractères spéciaux.
help	Affiche l'aide.
prompt	Suivi de on ou off, active ou désactive la confirmation individuelle de transfert pour chaque fichier (mget ou mput).
pwd	Affiche le répertoire distant courant.
cd	Suivi du chemin, déplacement dans l'arborescence distante.
ls	Liste les fichiers de la machine distante.
delete	Suivi d'un nom de fichier, supprime le fichier distant.
mdelete	Multiple. Supprime les fichiers distants.
get	Récupère le fichier distants.
mget	Multiple. Récupère les fichiers distants (liste ou modèle).
put	Envoie le fichier local vers la machine distante.
mput	Multiple. Envoie les fichiers locaux sur la machine distante (liste ou modèle).

Commande	Action
close/disconnect	Ferme la session actuelle.
lcd	Change de répertoire sur la machine locale.
hash	Durant les transferts, écrit un « # » sur écran pour chaque buffer transféré.
system	Informations sur le système distant.
recv	Réception d'un fichier.
send	Envoi d'un fichier.
rename	Renomme un fichier distant.
mkdir	Crée un répertoire sur la machine distante.
rmdir	Supprime un répertoire sur la machine distante.
!commande	Exécute la commande locale.

1.4.2 Telnet

Telnet est un client léger permettant d'ouvrir une connexion et une session sur une machine distante proposant un serveur telnet. Ce serveur est souvent lancé depuis xinetd ou inetd. Sa syntaxe est très simple :

```
$ telnet -l user machine port
Exemple :
# telnet 192.168.1.60
Trying 192.168.1.60...
Connected to 192.168.1.60.
Escape character is '^]'.
Welcome to openSUSE 10.3 (i586) - Kernel 2.6.24.4-default (3).

slyserver login: seb
Mot de passe :
Dernière connexion : jeu-
di 15 mai 2008 à 06:26:30 CEST de console
sur :0
Vous avez un nouveau message.
Have a lot of fun...
seb@slyserver:~>
```

■ Remarque

Attention ! Le service (et le client) telnet n'est absolument pas sécurisé : les connexions transitent en clair sur le réseau, et n'importe quel renifleur IP (wireshark par exemple) peut intercepter et voir tout ce qui est fait. Même le mot de passe est transmis en clair (en texte). Évitez d'utiliser ce service et concentrez-vous sur OpenSSH.

1.4.3 Ping

La commande **ping** est une commande centrale, voire incontournable. La première chose que l'on fait généralement pour savoir si une machine est accessible ou non, c'est d'essayer de la « pinguer » (sous réserve que la configuration du firewall autorise les requêtes ICMP). La commande **ping6** fait de même pour une adresse IPv6.

Ping émet un « écho » réseau, un peu comme un sonar, et attend une réponse, le retour de l'écho. Il utilise pour cela le protocole ICMP. Interrompez la commande ping avec [Ctrl] C.

```
$ ping www.kde.org
PING www.kde.org (62.70.27.118) 56(84) bytes of data.
64 bytes from 62.70.27.118: icmp_seq=1 ttl=57 time=10.5 ms
64 bytes from 62.70.27.118: icmp_seq=2 ttl=57 time=11.3 ms
64 bytes from 62.70.27.118: icmp_seq=3 ttl=57 time=10.4 ms
64 bytes from 62.70.27.118: icmp_seq=4 ttl=57 time=11.5 ms
...
...
```

Trois paramètres doivent attirer votre attention :

- -c permet de préciser le nombre d'échos à émettre.
- -b permet d'émettre un écho sur une adresse de broadcast.
- -I permet de spécifier l'interface réseau.

Dans le premier cas le paramètre peut être utile pour tester dans un script si un serveur répond :

```
# ping -c 1 10.9.238.170 >/dev/null 2>&1 && echo "Le serveur répond"
Le serveur répond
```

Dans le second cas, toutes les adresses du sous-réseau concerné par l'adresse de broadcast doivent répondre.

```
# ping -b 192.168.1.255
WARNING: pinging broadcast address
PING 192.168.1.255 (192.168.1.255) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.232 ms
64 bytes from 192.168.1.60: icmp_seq=1 ttl=64 time=0.240 ms
64 bytes from 192.168.1.130: icmp_seq=1 ttl=255 time=0.285 ms
64 bytes from 192.168.1.139: icmp_seq=1 ttl=255 time=0.292 ms
...
...
```

Dans le dernier cas, vous pouvez spécifier une carte de sortie. Cette option est très utile pour vérifier une résolution DNS ou une route.

```
# ping -I eth0 192.168.1.60
PING 192.168.1.60 (192.168.1.60) from 192.168.1.10:eth0: 56(84) bytes
of data.
```

```
| 64 bytes from 192.168.1.60: icmp_seq=1 ttl=62 time=0.478 ms
| 64 bytes from 192.168.1.60: icmp_seq=2 ttl=62 time=0.408 ms
| ...
```

1.4.4 Traceroute

Quand vous tentez d'accéder à un hôte distant depuis votre machine, les paquets IP passent souvent par de nombreuses routes, parfois différentes selon le point de départ et de destination, l'engorgement, etc. Le trajet passe par de nombreuses passerelles (gateways), qui dépendent des routes par défaut ou prédefinies de chacune d'elles.

La commande **traceroute** permet de visualiser chacun des points de passage de vos paquets IP à destination d'un hôte donné. Dans l'exemple suivant, l'hôte situé en région parisienne sur le réseau du fournisseur Free tente de déterminer la route empruntée pour se rendre sur le serveur www.kde.org. L'adresse IP source (hors réseau local) est volontairement masquée. La commande **traceroute6** est identique pour IPv6.

```
$ traceroute www.kde.org
traceroute to www.kde.org (62.70.27.118), 30 hops max, 40 byte packets
 1 DD-WRT (192.168.1.1)  0.558 ms   0.533 ms   0.585 ms
 2 82.xxx.yyy.zzz (82.xxx.yyy.zzz)  6.339 ms   6.404 ms   6.901 ms
 3 * * *
 4 * * *
 5 212.73.205.5 (212.73.205.5)  39.267 ms   35.499 ms   31.736 ms
 6 ae-12-55.car2.Paris1.Level3.net (4.68.109.144)  6.485 ms ae-22-
52.car2.Paris1.Level3.net (4.68.109.48)  6.401 ms   6.338 ms
 7 UUnet-Level13.Level13.net (212.73.240.206)  6.113 ms   6.152 ms
 5.866 ms
 8 so-3-2-0.TL2.PAR2.ALTER.NET (146.188.8.121)  6.107 ms   6.410 ms
 6.365 ms
 9 so-2-2-0.TL2.STK2.ALTER.NET (146.188.7.33)  87.323 ms   86.840 ms
 87.010 ms
10 so-7-1-0.XR2.OSL2.ALTER.NET (146.188.15.62)  96.491 ms   97.148 ms
 96.488 ms
11 ge-0-1-0.GW6.OSL2.ALTER.NET (146.188.3.242)  95.972 ms   95.934 ms
 96.108 ms
12 213.203.63.74 (213.203.63.74)  95.320 ms   94.321 ms   96.188 ms
13 leeloo.troll.no (62.70.27.10)  94.064 ms   94.052 ms   92.374 ms
14 jamaica.kde.org (62.70.27.118)  97.064 ms   96.182 ms   97.853 ms
```

1.4.5 Whois

Savez-vous que vous pouvez obtenir toutes les informations voulues sur un domaine (toto.fr) à l'aide de la commande **whois** ? Par exemple, pour obtenir toutes les informations sur le domaine kde.org :

```
| > whois kde.org
| ...
```

Domain ID:D1479623-LROR
Domain Name:KDE.ORG
Created On:14-Dec-1996 05:00:00 UTC
Last Updated On:12-Oct-2007 13:10:18 UTC
Expiration Date:13-Dec-2012 05:00:00 UTC
Sponsoring Registrar:easyDNS Technologies Inc. (R1247-LROR)
Status:CLIENT TRANSFER PROHIBITED
Status:CLIENT UPDATE PROHIBITED
Registrant ID:tu2YDGaiunEvz5QA
Registrant Name:Trolltech AS
Registrant Organization:Trolltech AS
Registrant Street1:Sandakerveien 116, PO Box 4332 Nydalen
Registrant Street2:
Registrant Street3:
Registrant City:Oslo
Registrant State/Province:N/A
Registrant Postal Code:N-0402
Registrant Country:NO
Registrant Phone:+1.4721604800
Registrant Phone Ext.:
Registrant FAX:+1.4721604801
Registrant FAX Ext.:
Registrant Email:hostmaster@trolltech.com
Admin ID:tubEUVkFFutkJZMD
Admin Name:Trolltech AS
Admin Organization:Trolltech AS
Admin Street1:Sandakerveien 116, PO Box 4332 Nydalen
Admin Street2:
Admin Street3:
Admin City:Oslo
Admin State/Province:N/A
Admin Postal Code:N-0402
Admin Country:NO
Admin Phone:+1.4721604800
Admin Phone Ext.:
Admin FAX:+1.4721604801
Admin FAX Ext.:
Admin Email:hostmaster@trolltech.com
Tech ID:tuSfxVVJtgMdkWm
Tech Name:Trolltech AS
Tech Organization:Trolltech AS
Tech Street1:Sandakerveien 116, PO Box 4332 Nydalen
Tech Street2:
Tech Street3:
Tech City:Oslo
Tech State/Province:N/A
Tech Postal Code:N-0402

```
| Tech Country:NO
```

```
| ...
```

1.4.6 Netstat

La commande **netstat** permet d'obtenir une foule d'informations sur le réseau et les protocoles.

Le paramètre **-i** permet d'obtenir l'état des cartes réseaux, afin de déterminer une éventuelle panne ou un problème de câble :

```
# netstat -i
Table d'interfaces noyau
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP
TX-OVR Flg
eth0 1500 0 2332007 0 0 0 677842 0 0
0 BMRU
lo 16436 0 1109 0 0 0 1109 0 0
0 LRU
```

Si vous rajoutez le paramètre **-e**, vous obtenez le même résultat qu'avec **ifconfig -a**.

```
# netstat -ei
Table d'interfaces noyau
eth0 Lien encap:Ethernet HWaddr 00:XX:D3:XX:AA:XX
      inet adr:12.168.1.60 Bcast:192.168.1.255 Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:2335314 errors:0 dropped:0 overruns:0 frame:0
          TX packets:678095 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:1055212145 (1006.3 Mb) TX bytes:61264196 (58.4 Mb)
          Interruption:20 Adresse de base:0x8c00

lo Lien encap:Boucle locale
    inet adr:127.0.0.1 Masque:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:1109 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1109 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 lg file transmission:0
        RX bytes:60423 (59.0 Kb) TX bytes:60423 (59.0 Kb)
```

Le paramètre **-r** permet d'obtenir, comme route, les tables de routage. Ajoutez le paramètre **-n** pour indiquer les IPs à la place des noms.

```
# netstat -rn
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic MSS Fenêtre irtt
Iface
192.168.211.0  0.0.0.0        255.255.255.0   U      0 0           0
vmnet8
```

192.168.1.0	0.0.0.0	255.255.255.0	U	0 0	0
eth0					
172.16.248.0	0.0.0.0	255.255.255.0	U	0 0	0
vmnet1					
169.254.0.0	0.0.0.0	255.255.0.0	U	0 0	0
eth0					
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0	0
lo					
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0	0
eth0					

Le paramètre **-a** permet de visualiser toutes les connexions, pour tous les protocoles, y compris les ports en écoute de la machine. La sortie est trop longue pour être placée dans ces pages.

```
# netstat -a | wc -l
495
```

Le paramètre **-A** permet de spécifier le protocole visible : **inet**, **inet6** (**ipv6**), **unix**, **ipx**, **ax25**, **netrom** et **ddp**.

```
# netstat -a -A inet
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale                               Adresse distante
Etat
tcp      0      0 localhost:716                                *:*
LISTEN
tcp      0      0 *:sunrpc                                 *:*
LISTEN
tcp      0      0 localhost:ipp                                *:*
LISTEN
tcp      0      0 localhost:smtp                                *:*
LISTEN
tcp      0      0 localhost:hpssd                                *:*
LISTEN
tcp      0      0 slyserver:41851                            imap.free.fr:imap
ESTABLISHED
tcp      0      0 slyserver:41850                            imap.free.fr:imap
ESTABLISHED
tcp      0      0 slyserver:54220                            by1msg4176111.gate:msnp
ESTABLISHED
tcp      0      0 slyserver:34267                            by2msg2105007.phx.:msnp
ESTABLISHED
tcp      0      0 slyserver:47990                            by1msg4082314.phx.:msnp
udp      0      0 *:filenet-tms                                *;*
udp      0      0 *:mdns                                    *;*
udp      0      0 *:sunrpc                                 *;*
udp      0      0 *:ipp                                    *;*
udp      0      0 172.16.248.1:ntp                           *;*
udp      0      0 192.168.211.1:ntp                           *;*
```

```
    | udp      0      0 slyserver:ntp          * : *
    | udp      0      0 localhost:ntp          * : *
    | udp      0      0 * : ntp              * : *
    | raw     0      0 * : icmp             * : *
```

7

Enfin le paramètre **-p** permet d'indiquer, quand c'est possible, le PID et le nom du processus.

```
# netstat -A inet -p
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale           Adresse distante
Etat      PID/Program name

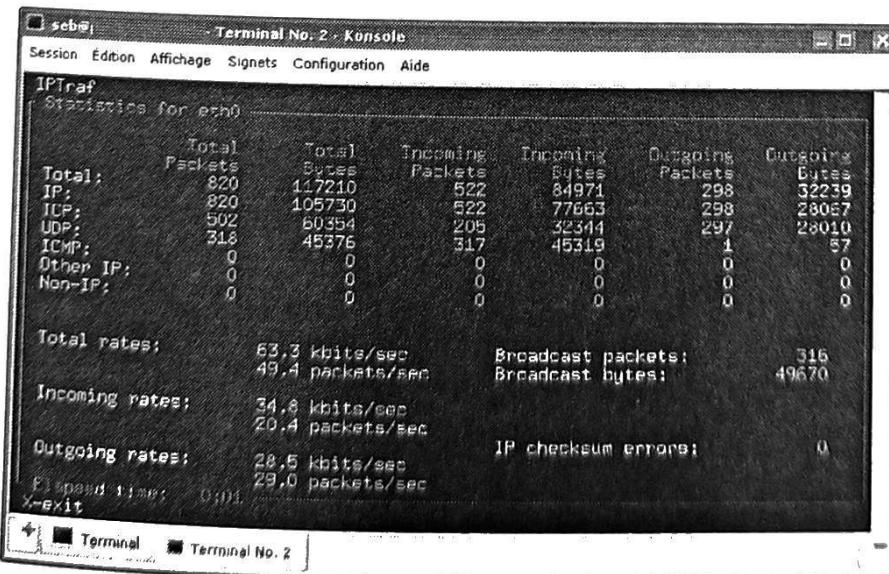
...
tcp      0      0 slyserver:54220          by1msg4176111.gate:msnp
ESTABLISHED 4041/kopete
tcp      0      0 slyserver:34267          by2msg2105007.phx.:msnp
ESTABLISHED 4041/kopete
tcp      0      0 slyserver:47990          by1msg4082314.phx.:msnp
ESTABLISHED 4041/kopete
```

1.4.7 IPtraf

La commande **iptraf** permet de visualiser en temps réel l'activité du réseau via un outil texte interactif ou non (ligne de commande). Les menus sont clairs. Vous vous y déplacez avec les touches fléchées et les divers raccourcis précisés.

La capture suivante montre l'affichage détaillé des statistiques de trafic de la carte eth0. Cet écran est accessible via la ligne de commande avec :

```
# iptraf -d eth0
```



IPtraf analyse le trafic sur eth0.

1.5 Fichiers généraux

1.5.1 /etc/resolv.conf

Le fichier **/etc/resolv.conf** est utilisé pour indiquer au système quels serveurs de noms et quels domaines interroger pour résoudre les requêtes DNS clientes. Les API sont incluses dans la bibliothèque et les API standards de Linux (il n'y a pas besoin d'ajouter des outils supplémentaires). On appelle cette bibliothèque le **resolver**.

■ Remarque

En configurant DHCP ce fichier est en principe mis automatiquement à jour et ne devrait pas être modifié sauf si vous avez interdit la configuration DNS sur votre client.

```
$ cat /etc/resolv.conf
domain mondomaine.org
search mondomaine.org
nameserver 192.168.1.1
nameserver 192.168.1.2
```

- **domain** : nom du domaine local. Les requêtes sont généralement réduites à des raccourcis relatifs au domaine local. S'il est absent le nom du domaine doit être déterminé à partir du nom d'hôte complet : c'est la partie située après le premier « . ».
- **search** : liste des domaines de recherche. Par défaut lors de l'utilisation de raccourcis (noms d'hôtes courts) le resolver lance une recherche sur le domaine défini par la ligne `domain`, mais on peut spécifier ici une liste de domaines séparés par des espaces ou des virgules.
- **nameserver** : adresse IP du serveur de noms (le serveur DNS). On peut en placer au maximum trois. Le resolver essaie d'utiliser le premier. En cas d'échec (timeout), il passe au second, et ainsi de suite.
- **options** : des options peuvent être précisées. Par exemple `timeout:n` où `n` (en secondes) indique le délai d'attente de réponse d'un serveur de noms avant de passer au suivant.

1.5.2 /etc/hosts et /etc/networks

Sans même utiliser de serveur de noms, vous pouvez établir une correspondance entre les adresses IP et les noms des machines au sein du fichier **/etc/hosts**.

```
192.168.1.1      server1 www1 ftp
192.168.1.11    poste1
192.168.1.12    poste2
```

Vous pouvez faire de même pour nommer les réseaux (ce qui peut être utile pour les `tcp_wrappers` ou la commande `route`) dans le fichier **/etc/networks**.

```
loopnet      127.0.0.0
localnet     192.168.1.0
```

1.5.3 /etc/nsswitch.conf

Le fichier **/etc/nsswitch.conf** permet de déterminer l'ordre dans lequel le resolver (ou d'autres services) récupère ses informations. Les deux lignes en gras de l'exemple indiquent que lors d'une requête de résolution de nom (ou de réseau) les fichiers sont prioritaires. Le fichier **/etc/hosts** est d'abord lu, puis, si le resolver ne trouve pas l'information il passe par une résolution DNS.

```
passwd:    compat
group:     compat

hosts:          files dns
networks:       files dns

services:    files
protocols:   files
rpc:         files
ethers:      files
netmasks:    files
netgroup:    files nis
publickey:   files

bootparams:  files

automount:    files nis
aliases:      files
```

Il peut arriver que certains produits mal programmés n'utilisent pas le resolver mais directement le DNS, ou le fichier `/etc/hosts`, ou inversent l'ordre établi dans `/etc/nsswitch.conf`. Dans ce cas, il n'est pas possible de prévoir (et de prédire) le bon fonctionnement de ce genre de produits...

1.5.4 /etc/services

Le fichier **/etc/services** contient la liste des services réseaux connus de Unix ainsi que les ports et protocoles associés. Il est utilisé par de nombreux services (dont `xinetd`) et sous-systèmes comme le firewall de Linux.

Ce fichier est indicatif : c'est un fichier de description et de définition : tous les services de ce fichier ne tournent pas forcément sur votre machine (et heureusement vu le nombre) ! Et un service peut être configuré pour écouter un autre port.

Par contre il est conseillé, lorsque vous rajoutez un service qui n'est pas présent dans ce fichier, de le rajouter à la fin.

```

tcpmux          1/tcp      # TCP Port Service Multiplexer
tcpmux          1/udp      # TCP Port Service Multiplexer
compressnet     2/tcp      # Management Utility
compressnet     2/udp      # Management Utility
compressnet     3/tcp      # Compression Process
compressnet     3/udp      # Compression Process
rje             5/tcp      # Remote Job Entry
rje             5/udp      # Remote Job Entry
echo            7/tcp      Echo
echo            7/udp      Echo
discard         9/tcp      # Discard
discard         9/udp      # Discard
systat          11/tcp     users      # Active Users
systat          11/udp     users      # Active Users
daytime         13/tcp     # Daytime (RFC 867)
daytime         13/udp     # Daytime (RFC 867)
netstat          15/tcp     # Unassigned [was netstat]
qotd            17/tcp     quote      # Quote of the Day
qotd            17/udp     quote      # Quote of the Day
msp              18/tcp     # Message Send Protocol
msp              18/udp     # Message Send Protocol
chargen          19/tcp     # Character Generator
chargen          19/udp     # Character Generator
ftp-data         20/tcp     # File Transfer [Default Data]
ftp-data         20/udp     # File Transfer [Default Data]
ftp              21/tcp     # File Transfer [Control]
fsp              21/udp     # File Transfer [Control]
ssh              22/tcp     # SSH Remote Login Protocol
ssh              22/udp     # SSH Remote Login Protocol
telnet           23/tcp     # Telnet
telnet           23/udp     # Telnet
...

```

1.5.5 /etc/protocols

Le fichier **/etc/protocols** contient la liste des protocoles connus par Unix.

# Assigned Internet Protocol Numbers				References
# Decimal	Keyword	Protocol	-----	
# protocol	num aliases	# comments	-----	-----
hopopt	0 HOPOPT	# IPv6 Hop-by-Hop Option		[RFC1883]
icmp	1 ICMP	# Internet Control Message		[RFC792]
igmp	2 IGMP	# Internet Group Management		[RFC1112]
ggp	3 GGP	# Gateway-to-Gateway		[RFC823]

ip	4	IP	# IP in IP (encapsulation)	[RFC2003]
st	5	ST	# Stream	[RFC1190, RFC1819]
tcp	6	TCP	# Transmission Control	[RFC793]
cbt	7	CBT	# CBT	[Ballardie]
egp	8	EGP	# Exterior Gateway Protocol	[RFC888, DLM1]
igp	9	IGP	# any private interior gateway	[IANA]
bbn-rcc-mon	10	BBN-RCC-MON	# BBN RCC Monitoring	[SGC]
nvp-ii	11	NVP-II	# Network Voice Protocol	[RFC741, SC3]
pup	12	PUP	# PUP	[PUP, XEROX]
argus	13	ARGUS	# ARGUS	[RWS4]
emcon	14	EMCON	# EMCON	[BN7]
xnet	15	XNET	# Cross Net Debugger	[IEN158, JFH2]
chaos	16	CHAOS	# Chaos	[NC3]
udp	17	UDP	# User Datagram	[RFC768, JBP]
...				

2. Services réseaux xinetd

2.1 Présentation

Le démon **xinetd** est un « super-service » permettant de contrôler l'accès à un ensemble de services, **telnet** par exemple. Beaucoup de services réseaux peuvent être configurés pour fonctionner avec xinetd, comme les services **ftp**, **ssh**, **samba**, **rccp**, **http**, etc. Des options de configuration spécifiques peuvent être appliquées pour chaque service géré.

Lorsqu'un hôte client se connecte à un service réseau contrôlé par xinetd, xinetd reçoit la requête et vérifie tout d'abord les autorisations d'accès TCP (voir **tcp_wrappers** au prochain chapitre) puis les règles définies pour ce service (autorisations spécifiques, ressources allouées, etc.). Une instance du service est alors démarrée et lui cède la connexion. À partir de ce moment **xinetd** n'interfère plus dans la connexion entre le client et le serveur.

2.2 Configuration

Les fichiers de configuration sont :

- **/etc/xinetd.conf** : configuration globale.
- **/etc/xinetd.d/*** : répertoire contenant les fichiers spécifiques aux services. Il existe un fichier par service, du même nom que celui précisé dans **/etc/services**.

```
$ ls -l /etc/xinetd.d
total 92
-rw-r--r-- 1 root root 313 sep 22 2007 chargen
-rw-r--r-- 1 root root 333 sep 22 2007 chargen-udp
-rw-r--r-- 1 root root 256 mar 20 22:11 cups-lpd
-rw-r--r-- 1 root root 409 nov  4 2005 cvs
-rw-r--r-- 1 root root 313 sep 22 2007 daytime
```

```
-rw-r--r-- 1 root root 333 sep 22 2007 daytime-udp
-rw-r--r-- 1 root root 313 sep 22 2007 discard
-rw-r--r-- 1 root root 332 sep 22 2007 discard-udp
-rw-r--r-- 1 root root 305 sep 22 2007 echo
-rw-r--r-- 1 root root 324 sep 22 2007 echo-udp
-rw-r--r-- 1 root root 492 sep 22 2007 netstat
-rw-r--r-- 1 root root 207 avr 23 19:04 rsync
-rw-r--r-- 1 root root 337 fév 17 14:22 sane-port
-rw-r--r-- 1 root root 332 sep 22 2007 servers
-rw-r--r-- 1 root root 334 sep 22 2007 services
-rw-r--r-- 1 root root 351 jun 21 2007 svnserve
-rw-r--r-- 1 root root 277 nov 8 2007 swat
-rw-r--r-- 1 root root 536 sep 21 2007 systat
-rw-r--r-- 1 root root 387 fév 4 10:11 tftp.rpmsave
-rw-r--r-- 1 root root 339 sep 22 2007 time
-rw-r--r-- 1 root root 333 sep 22 2007 time-udp
-rw-r--r-- 1 root root 2304 avr 4 11:39 vnc
-rw----- 1 root root 768 sep 22 2007 vsftpd
```

Contenu de xinetd.conf :

```
defaults
{
    instances          = 60
    log_type           = SYSLOG authpriv
    log_on_success     = HOST PID
    log_on_failure     = HOST
    cps                = 25 30
}
includedir /etc/xinetd.d
```

- **instances** : nombre maximal de requêtes qu'un service xinetd peut gérer à un instant donné.
- **log_type** : dans notre cas, les traces sont gérées par le démon **syslog** via **authpriv** et les traces sont placées dans **/var/log/secure**. **FILE /var/log/xinetd** aurait placé les traces dans **/var/log/xinetd**.
- **log_on_success** : xinetd va journaliser l'événement si la connexion au service réussit. Les informations tracées sont l'hôte (**HOST**) et le **PID** du processus serveur traitant la connexion.
- **log_on_failure** : idem mais pour les échecs. Il devient simple de savoir quels hôtes ont tenté de se connecter si par exemple la connexion n'est pas autorisée.
- **cps** : xinetd n'autorise que 25 connexions par secondes à un service. Si la limite est atteinte, xinetd attendra 30 secondes avant d'autoriser à nouveau les connexions.
- **includedir** : inclut les options des fichiers présents dans le répertoire indiqué.

Exemple /etc/xinetd.d/telnet :

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
#               unencrypted username/password pairs for authentication.
service telnet
{
    disable = no
    flags     = REUSE
    socket_type = stream
    wait      = no
    user      = root
    server    = /usr/sbin/in.telnetd
    log_on_failure += USERID
}
```

La première ligne en commentaire, **default**, a une importance particulière. Elle n'est pas interprétée par xinetd mais par **ntsysv** ou **chkconfig** pour déterminer si le service est actif.

- **service** : nom du service qui correspond à un service défini dans /etc/services.
- **flags** : attributs pour la connexion. REUSE indique que la socket sera réutilisée pour une connexion telnet.
- **socket_type** : spécifie le type de socket. Généralement **stream** (tcp) ou **dgram** (udp). Une connexion directe IP se fait par **raw**.
- **wait** : indique si le serveur est single-threaded (yes) ou multi-threaded (no).
- **user** : sous quel compte utilisateur le service sera lancé.
- **server** : chemin de l'exécutable devant être lancé.
- **log_on_failure** : le **+=** indique qu'on rajoute l'option associée au fichier de trace en plus de celles par défaut. Ici : le login.
- **disable** : indique si le service est actif ou non.

Certaines options peuvent améliorer les conditions d'accès et la sécurité :

- **only_from** : permet l'accès uniquement aux hôtes spécifiés.
- **no_access** : empêche l'accès aux hôtes spécifiés (ex : 172.16.17.0/24).
- **access_times** : autorise l'accès uniquement sur une plage horaire donnée (ex : 09:00-18:30).

2.3 Démarrage et arrêt des services

On distingue deux cas.

Premier cas, le service **xinetd** est un service comme un autre dont le démarrage ou l'arrêt peut s'effectuer avec la commande **service** ou directement via l'exécution de /etc/init.d/xinetd.

```
| # service xinetd start
```

Dans ce cas, la commande **chkconfig** (Red Hat, openSUSE) autorise ou non le lancement du service au démarrage pour chaque niveau d'exécution (runlevel).

```
| # chkconfig --level 345 xinetd on
```

Second cas, comme xinetd gère plusieurs services, l'arrêt de xinetd arrête tous les services associés, et le démarrage de xinetd lance tous les services associés. Il n'est pas possible de choisir quels services de xinetd seront lancés dans tel ou tel niveau d'exécution. Mais vous pouvez choisir d'activer ou de désactiver simplement un service avec chkconfig.

```
| # chkconfig telnet on
```

3. Connexion PPP

3.1 Choix et réglage du modem

3.1.1 Le cas des Winmodems

Tous les modems RTC (analogiques) se connectant sur un port série (externe), ou émulant un vrai port série (carte PCI ou via le port USB) sont entièrement supportés sous Linux.

Cependant, il existe une catégorie particulière de modems appelés les **winmodems**. Ils se présentent parfois comme des « vrais » modems (ils leur ressemblent parfois). D'une manière générale, fuyez ce type de modems. Cependant quelques modèles sont connus pour fonctionner sous Linux. Rendez-vous sur le site <http://linmodems.org/> pour obtenir des informations à ce sujet.

D'autres adaptateurs que les modems RTC sont reconnus par Linux comme des modems ; c'est le cas de quelques adaptateurs ADSL, mais aussi des téléphones portables reliés par une câble USB ou via une connexion Bluetooth.

3.1.2 Les fichiers périphériques

Les ports série de type RS232 se nomment **ttySn** :

- **/dev/ttys0** : premier port série.
- **/dev/ttys1** : second port série.
- etc.

Les ports série de type USB se nomment **ttyUSBn** : **/dev/ttys0**, et ainsi de suite.

Les ports de communication série via bluetooth se nomment **rfcommn** (pour radio frequency communication) : **/dev/rfcomm0**, et ainsi de suite.

Pour utiliser les ports série et établir une communication, vous devez pouvoir écrire sur les périphériques (pour envoyer les ordres) et donc soit avoir les droits correspondants, soit utiliser un programme SUID, ou encore disposer de règles udev adaptées.

3.1.3 Régler le port série

Les ports série se gèrent via la commande **setserial**.

La commande **setserial** permet d'interroger la configuration d'un port série avec le paramètre **-g**. Le port série **ttyS0** est de type **16550A** (le plus rapide), utilise l'IRQ **4** et le port d'adresse **0x03f8**.

```
# setserial -g /dev/ttyS0  
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
```

Les informations n'étant pas suffisantes, rajoutez le paramètre **-a**. Vous obtenez entre autres la vitesse de la ligne qui est de **115200** bits par seconde (le mot baud est à proscrire).

```
# setserial -a -g /dev/ttyS0  
/dev/ttyS0, Line 0, UART: 16550A, Port: 0x03f8, IRQ: 4  
      Baud_base: 115200, close_delay: 50, divisor: 0  
      closing_wait: 3000  
      Flags: spd_normal skip_test
```

Setserial permet aussi de configurer le port. En interrogeant le port, vous pouvez savoir quels paramètres ont permis son réglage avec le **-G** :

```
# setserial -G /dev/ttyS0  
/dev/ttyS0 uart 16550A port 0x03f8 irq 4 baud_base 115200 spd_normal  
skip_test
```

De là, il est possible d'extrapoler de nouvelles valeurs. Par exemple passez ainsi le port série à une vitesse de **57600 bps** :

```
# setserial /dev/ttyS0 baud_base 57600  
# setserial -a -g /dev/ttyS0  
/dev/ttyS0, Line 0, UART: 16550A, Port: 0x03f8, IRQ: 4  
      Baud_base: 57600, close_delay: 50, divisor: 0  
      closing_wait: 3000  
      Flags: spd_normal skip_test
```

3.1.4 Les commandes AT

Les modems utilisent tous un jeu de commandes standard appelées **commandes AT**. Leur vrai nom est **Commandes Hayes**, du nom de la société les ayant inventées. AT signifie Attention. Le modem attend après ces premières lettres une suite permettant de le configurer, de numéroter, de raccrocher, etc.

Le jeu est en principe standard mais la configuration varie d'un modèle à un autre. Si vous ne rentrez pas dans les détails, une configuration générique suffit et fonctionne pour la quasi-totalité des modems. Comme il n'est pas possible de décrire les commandes AT ici, vous en trouverez une liste sur Wikipedia : http://fr.wikipedia.org/wiki/Commandes_Hayes.

En voici tout de même quelques-unes :

- Numéroter : ATDT0102030405
- Répondre : ATA
- Raccrocher : ATH

3.2 PPP

Le protocole **PPP** (*Point to Point Protocol*) permet de vous relier à une autre machine afin d'y accéder, ou à son réseau, ce qui est souvent le cas d'Internet. Aujourd'hui encore et malgré les nombreuses solutions proposées par le câble ou l'ADSL, certaines connexions se font encore via un modem RTC (modem classique connecté sur port USB, série ou interne) relié à une prise téléphonique classique.

L'établissement d'une liaison PPP nécessite :

- Un client disposant des outils ppp (pppd) et chat pour dialoguer avec le serveur.
- Un serveur disposant de pppd et des moyens de fournir une adresse IP (dhcp).
- Un modem.

La suite ne prend en considération que la partie cliente.

Vous devez connaître :

- Le port série sur lequel est branché votre modem : ttySX (série ou USB), ttyACMX (USB), rfcommX (Bluetooth), etc.
- Le numéro d'appel de votre fournisseur d'accès Internet (FAI).
- Le nom d'utilisateur et le mot de passe chez votre FAI.
- L'adresse du serveur DNS de votre FAI.

■ Remarque

Le modem n'est pas forcément RTC. Un téléphone portable reconnu comme modem via le câble de connexion au PC ou depuis le protocole Bluetooth fait un excellent modem. Pour peu qu'il soit aux normes 3G ou Edge, les débits peuvent être très impressionnantes.

■ Remarque

Un grand nombre de connexions ADSL sont aussi effectuées via le protocole PPP. Dans ce cas la suite s'applique mais des modifications sont à prévoir.

3.3 Connexion via la console

3.3.1 À la main

Dans l'exemple qui suit :

- Le numéro de téléphone est 0102030405.
- Le login est « login ».
- Le mot de passe est « password ».
- Le périphérique est /dev/modem.

Il peut être nécessaire, bien que cela puisse être géré par DHCP au moment de la connexion, de modifier le fichier /etc/resolv.conf pour indiquer les serveurs de noms (DNS) de votre fournisseur.

La connexion PPP nécessite que le service **pppd** (généralement /usr/sbin/pppd) soit exécuté en tant que root. Pour cela, le droit SUID est souvent positionné :

```
# -rwsr-xr-t 1 root root 316392 2008-04-04 19:03 pppd*
```

Une autre solution est de donner ces droits à l'outil de connexion (chat, kppp, etc.) ou de modifier en conséquence les droits des périphériques (cas de plusieurs distributions).

Les fichiers de configuration sont situés dans /etc/ppp :

```
# ls -l /etc/ppp
total 48
-rw----- 1 root root 690 sep 21 2007 chap-secrets
-rw-r--r-- 1 root root 449 sep 21 2007 filters
lrwxrwxrwx 1 root root 5 mai 9 20:47 ip-down -> ip-up
drwxr-xr-x 2 root root 4096 sep 21 2007 ip-down.d
-rw xr-xr-x 1 root root 6175 avr 24 00:26 ip-up
drwxr-xr-x 2 root root 4096 sep 21 2007 ip-up.d
-rw-r--r-- 1 root root 7943 sep 21 2007 options
-rw----- 1 root root 340 sep 21 2007 options.pptp
-rw----- 1 root root 1219 sep 21 2007 pap-secrets
drwxr-xr-x 2 root root 4096 fév 23 23:18 peers
-rw xr-xr-x 1 root root 3778 avr 24 00:26 poll.tecip
```

Voici un exemple de connexion à un serveur PPP :

```
#!/bin/sh
/usr/sbin/pppd connect '/usr/sbin/chat -v ABORT ERROR ABORT "NO'
```

```
CARRIER" \
ABORT BUSY "" ATZ OK ATDT0102030405 CONNECT "" ogin: "login" \
word: "password" \
/dev/modem 38400 noipdefault debug crtscts modem defaultroute &
```

3.3.2 Par les fichiers

Vous allez avoir besoin de deux fichiers. Le premier va contenir les commandes du service pppd, le second la séquence de communication avec le FAI. Les deux sont placés dans /etc/ppp/peers.

Soit le premier fichier /etc/ppp/peers/cnx1 :

```
# cat /etc/ppp/peers/cnx1
/dev/modem
connect '/usr/sbin/chat -v -f /etc/ppp/peers/cnx1-chat'

defaultroute
noipdefault
usepeerdns
115200

debug
noauth

maxfail 10
lcp-echo-interval 5
lcp-echo-failure 12
holdoff 3
noaccomp noccp nobsdcomp nodeflate nopcomp novj novjccomp
lock
crtscts
```

Chaque ligne contient au moins une instruction dont voici les plus pertinentes :

- **/dev/modem** : le périphérique de connexion (le modem).
- **connect** : la chaîne de connexion envoyée au FAI.
- **defaultroute** : la route par défaut est remplacée par celle fournie par le FAI.
- **noipdefault** : le FAI fournit l'IP par son DHCP.
- **usepeerdns** : récupère les informations DNS du FAI.
- **115200** : la vitesse de communication du périphérique (elle sera négociée).
- **debug** : fournit le détail complet de la connexion.
- **noauth** : ce n'est pas le script ppp qui établit l'authentification (voir ligne connect).
- **maxfail** : n tentatives de connexion avant d'abandonner.
- **holdoff** : attente de n secondes entre deux connexions.

- **lock** : permet l'accès exclusif au fichier périphérique.
- **crtscts** : active le contrôle de flux matériel.

Soit le second fichier `/etc/ppp/peers/cnx1-chat` utilisé par la ligne **connex** du premier fichier `/etc/ppp/peers/cnx1` :

```
# cat /etc/ppp/peers/cnx1-chat
ABORT ERROR
ABORT "NO CARRIER"
ABORT BUSY "" ATZ
OK ATDT0102030405
CONNECT ""
login: "login"
word: "password"
```

Il ne s'agit que d'un exemple. Vous devez vérifier tant du côté de votre FAI que du côté de la documentation de votre modem quelles sont les bonnes commandes AT à passer (elles sont généralement standard).

3.3.3 Connexion

Initialisez la connexion :

```
# pppd call cnx1
```

Vous devriez voir les leds de votre modem clignoter, et si le haut-parleur est activé le bruit caractéristique se fait entendre. Si la connexion est établie, une nouvelle interface réseau apparaît : **ppp0**.

```
# ifconfig ppp0
ppp0      Link encap:Point-Point Protocol
          inet addr:10.xx.yy.zz  P-t-P:10.xx.yy.zz Mask:255.255.255.0
                  UP POINTOPOINT RUNNING MTU:552 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0
                  TX packets:0 errors:0 dropped:0 overruns:0
```

■ Remarque

Les distributions sont souvent accompagnées d'outils de type NetworkManager qui permettent une connexion ethernet, Wi-Fi ou ppp à la volée en quelques clics. La distribution Mandriva mérite une mention spéciale : la configuration d'une connexion Internet via un mobile 3G au travers du protocole Bluetooth s'est fait en quelques secondes, l'outil Drakconf ayant tout détecté seul.

4. OpenSSH

4.1 Présentation

OpenSSH est un protocole de shell sécurisé, un mécanisme qui permet l'authentification sécurisée, l'exécution à distance et la connexion à distance. Il permet aussi le transport sécurisé du protocole X Window. Enfin, il est capable d'encapsuler des protocoles non sécurisés en redirigeant les ports.

Les packages à utiliser pour un serveur sont **openssh**, **openssl** et **openssh-clients**. Pour X on rajoute les packages **openssh-askpass*** (il peut y en avoir plusieurs suivant l'environnement de bureau). La liste des packages à installer dépend de chaque distribution.

L'utilisation la plus commune reste l'accès distant sécurisé à une machine via le client ssh.

4.2 Configuration

La configuration est `/etc/ssh/sshd_config`. Quelques options sont éventuellement à modifier :

- **Port** : le numéro de port, par défaut 22.
- **Protocol** : fixé à 2,1 il autorise SSH1 et SSH2. On préférera SSH2 et donc on laissera la valeur 2 seule.
- **ListenAddress** : par défaut ssh écoute sur toutes les IP du serveur. On peut autoriser uniquement l'écoute sur une interface donnée.
- **PermitRootLogin** : ssh autorise les connexions de root. On peut placer la valeur à « **no** ». Dans ce cas, il faudra se connecter en simple utilisateur et passer par **su** ou **sudo**.
- **Banner** : chemin d'un fichier dont le contenu sera affiché aux utilisateurs lors de la connexion.

Ssh est un service System V à lancer avec `service` ou directement par `/etc/init.d/sshd`.

```
# service sshd start
```

4.3 Utilisation

La commande **ssh** permet d'établir une connexion.

```
$ ssh -l login host
$ ssh login@host
```

L'option **-X** permet d'activer la redirection (**forwarding**) du protocole X Window.

```
| $ ssh -X login@host
```

4.4 Clés et connexion automatique

Il est possible d'établir une connexion automatique vers une autre machine sans saisir de mot de passe. Pour cela, il est nécessaire depuis le compte utilisateur du client (la machine qui va se connecter) de générer une paire de clés, privée et publique. Aucune passphrase ne doit être saisie.

Du côté du serveur ssh, la clé publique du client doit être placée dans un fichier contenant les clés autorisées à se connecter dans le compte de destination.

4.4.1 Côté client

- Générez une clé au format RSA avec la commande **ssh-keygen** :

```
| $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bean/.ssh/id_rsa):
Created directory '/home/bean/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bean/.ssh/id_rsa.
Your public key has been saved in /home/bean/.ssh/id_rsa.pub.
The key fingerprint is:
f6:39:23:4e:fa:53:d0:4e:65:7f:3f:fd:a3:f4:8e:2a bean@p64p17bicb3
```

- Le répertoire de l'utilisateur contient maintenant un répertoire .ssh :

```
| $ cd .ssh
| $ ls
| id_rsa  id_rsa.pub
```

- Le fichier id_rsa.pub contient la clé publique :

```
| $ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEArcB/VskR9v708J2EDG1LM1Q6HmKJC
P2UenurnSr7rWTSZK5w9Hzn4DCz5iMzLAPc4659I0uKJbmF3vBXozIgLrCdCZCQE
hhPLwJVLXbGNc81Mf742E/WqkkJ/uQYb31iPAU7Efosei+DVZ21No725XjiSCZ2q
zKKx7ZuNQEtW0eVkwv1A0u7Hvrwn+FQksW3NXwTxwHhudSw7S6kIC3tyF5rkzf
vu7zQbOGDGGPiF3aOvd0oSBNgijTz+M0PaoXXI3brMd6WkGfSwf4oFYKNDC/3T
Q4xU6WxkxqTBcsjEm1gIymFAyxDo+zzf63jxLGO8Pp50DKf7DUqBx7+rjw==
bean@slyserver
```

4.4.2 Côté serveur

- Allez dans le répertoire .ssh du compte auquel vous souhaitez accéder sur le serveur (créez-le s'il n'existe pas).

■ \$ cd /home/seb/.ssh

- Éditez le fichier authorized_keys2 (créez-le s'il n'existe pas) et copiez-y sur une nouvelle ligne le contenu du fichier id_rsa.pub du client. Sauvez.

```
$ echo "ssh-rsa
AAAAB3NzaC1yc2EAAAQEA... /VskR9v708J2EDG1LM1Q6HmKJcP2Uenurn
Sr7rWTSZK5w9Hzn4DCz5iMzLAPc4659I0uKJbmF3vBXozIgLrCdCZCQEhhPLwJVL
XbGNc81Mf742E/WqkkJ/uQYb31iPAU7Efosei+DVZ21No725XjiSCZ2qzKKx7ZuN
QEtxW0eVkwv1A0u7Hvrwn+FQksW3NXwTxwHhudSw7S6kIC3tyF5rkzfkvu7zQbOG
DGGPiF3aOvd0oSBNgijtZ+M0PaoXXI3brMd66WkGfSwf4ofYKNDCA/3TQ4xU6Wxk
xqTBcsjEm1gIymFAyxDo+zzf63jxLGO8Pp50DKf7DUqBx7+rjw== bean@slyser
ver" >> authorized_keys2
```

- Tentez une connexion, le mot de passe n'est pas demandé :

■ \$ ssh seb@slyserver

5. Monter un serveur DHCP

5.1 Présentation

Le service **DHCP** (*Dynamic Host Configuration Protocol*), protocole de configuration dynamique des hôtes, permet aux hôtes d'un réseau de demander et recevoir des informations de configuration (adresse, routage, DNS, etc.). Il y a en général un seul serveur DHCP par segment de réseau même si plusieurs sont possibles. Si le serveur est sur un autre segment, on peut utiliser un agent de retransmission DHCP.

Autrement dit, un client DHCP recherche tout seul un serveur DHCP qui lui communiquera son adresse IP. L'adresse IP est assignée soit dynamiquement à partir de plages d'adresses prédéfinies, soit statiquement en fonction de l'adresse MAC du demandeur. Les informations sont valables un laps de temps donné (un bail) qui peut être renouvelé et configurable.

DHCP est un sur-ensemble de **BOOTP** (*Bootstrap Protocol*). Quand le client cherche à contacter un serveur, c'est BOOTP qui fournit les informations d'adressage. DHCP gère les renouvellements. BOOTP se base sur le protocole de transport UDP.

Un hôte n'a aucune information réseau disponible au démarrage. Il doit trouver seul un serveur DHCP. Pour cela, BOOTP effectue un broadcast sur l'IP 255.255.255.255 avec une trame contenant ses informations (comme son adresse MAC) et les informations souhaitées (type de requête, ici DHCPDISCOVER, port de connexion, etc.). Le broadcast est envoyé par définition à tous les hôtes du réseau local. Quand le serveur DHCP détecte la trame, il effectue lui aussi un

broadcast (l'hôte client n'a pas encore d'IP) avec les informations de base souhaitées par l'hôte (DHCPoffer, premiers paramètres). L'hôte établit une première configuration puis demande confirmation de l'IP (DHCPrequest). Le serveur DHCP confirme (DHCPack). Le bail est confirmé et le client dispose dès lors de toutes les informations valides.

5.2 Serveur dhcpd

5.2.1 Démarrage

Le serveur **dhcpd** est un service (daemon) lancé à l'aide d'un script (/etc/init.d/dhcpd). Il est configuré à l'aide du fichier /etc/dhcpd.conf. Les adresses IP allouées sont placées dans /var/lib/dhcp/dhcpd.leases.

```
| # service dhcpcd start
```

Ou :

```
| # /etc/init.d/dhcpcd start
```

5.3 Informations de base

Le fichier de configuration d'un serveur est, si vous restez dans des réglages de base, assez simple.

```
ddns-update-style none; # pas de mise à jour du DNS par DHCP
option domain-name "toto.fr"; # nom de domaine transmis au client
option domain-name-servers 192.168.1.254; # liste des DNS séparés
par des virgules
default-lease-time 21600; # durée du bail par défaut en secondes
sans demande explicite
max-lease-time 43200; # durée max du bail si la demande du client
est plus élevée
```

Comme dhcpcd peut gérer plusieurs sous-réseaux, on doit lui préciser les règles à appliquer pour chaque sous-réseau. Généralement dans le cadre d'un petit réseau un seul bloc sera présent mais tous les cas sont envisageables. Si vous êtes certain de n'avoir qu'un seul réseau, vous pouvez omettre la déclaration du subnet (sous-réseau).

```
# Gestion du sous-réseau 192.168.1.0
subnet 192.168.1.0 netmask 255.255.255.0
{
    option routers 192.168.1.254; # passerelle pour ce réseau
    option subnet-mask 255.255.255.0; # masque de sous-réseau
    range 192.168.1.2 192.168.1.250; # Configuration de l'intervalle DHCP
```

```
# Cas d'attributions d'IP statiques
host station1
{
    hardware ethernet 00:A0:ad:41:5c:b1; # Adresse MAC
    fixed-address 192.168.1.1; # cette machine aura l'IP 192.168.1.1
}
```

■ Remarque

Certains clients DHCP ignorent totalement le fait qu'un serveur DHCP peut dynamiquement allouer un nom (*hostname*) à l'hôte. Dans l'exemple précédent, la machine avec l'IP 192.168.1.1 devrait obtenir le nom *station1*. Voici un exemple :

```
# les hôtes se verront attribuer les noms des host déclarés
use-host-decl-names on;
host station1
{
    hardware ethernet 00:A0:ad:41:5c:b1; # Adresse MAC
    fixed-address 192.168.1.1; # cette machine aura l'IP 192.168.1.1
et le nom station1
}
```

Vous pouvez aussi travailler au cas par cas :

```
host station2
{
    hardware ethernet 00:A0:ad:41:5c:b2; # Adresse MAC
    fixed-address 192.168.1.251; # ce host aura l'IP 192.168.1.251
    option host-name "station2"; # ce host aura comme nom station2
}
```

5.4 Côté client

Sous Linux et les distributions de type Red Hat, Fedora, Mandriva, openSUSE, etc., modifiez le fichier **/etc/sysconfig/network-script/ifcfg-xxx** en spécifiant **dhcp** pour **BOOTPROTO** et relancez la connexion réseau (**ifdown** puis **ifup**).

Le client **dhpcd** permet d'activer dhcp sur une interface réseau. La méthode la plus simple est, par exemple pour **eth0** :

```
# dhpcd eth0 &
```

Vous pouvez transmettre des options à **dhpcd** pour prendre en charge diverses possibilités. Parmi ces options :

- **-D** : autorise la modification du nom de domaine.
- **-H** : autorise la modification du nom d'hôte.
- **-R** : évite l'écrasement du fichier **resolv.conf**.

- -l : permet de modifier le leasetime (en secondes).

```
# dhcpcd -D -H -l 86400 eth0
```

6. Serveur DNS

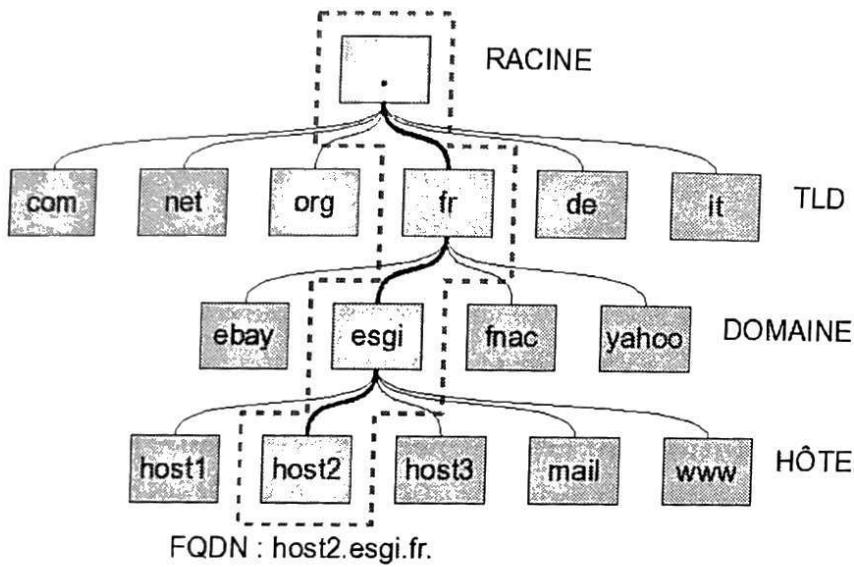
6.1 Présentation

Le Système de Noms de Domaine **DNS** (*Domain Name System*) transforme les noms d'hôte en adresses IP : c'est la **Réolution de nom**. Il transforme les adresses IP en noms d'hôte : c'est la **Réolution inverse**. Il permet de regrouper les machines par domaines de nom. Il fournit des informations de routage et de courrier électronique.

Le DNS permet de faire référence à des systèmes basés sur IP (les *hôtes*) à l'aide de noms conviviaux (les *noms de domaine*). L'intérêt d'un DNS est évident. Les noms de domaine sont plus simples à retenir, et si son adresse IP change l'utilisateur ne s'en rend même pas compte. On comprend que le DNS est un service clé critique pour Internet.

Les noms de domaine sont séparés par des points, chaque élément pouvant être composé de 63 caractères ; il ne peut y avoir qu'un maximum de 127 éléments et le nom complet ne doit pas dépasser 255 caractères. Le nom complet non abrégé est appelé **FQDN** (*Fully Qualified Domain Name*). Dans un FQDN, l'élément le plus à droite est appelé **TLD** (*Top Level Domain*), celui le plus à gauche représente l'hôte et donc l'adresse IP.

Le DNS contient une configuration spéciale pour les routeurs de courrier électronique (définitions MX) permettant une résolution inverse, un facteur de priorité et une tolérance de panne.



Représentation d'une arborescence DNS

Une zone est une partie d'un domaine gérée par un serveur particulier. Une zone peut gérer un ou plusieurs sous-domaines, et un sous-domaine peut être réparti en plusieurs zones. Une zone représente l'unité d'administration dont une personne peut être responsable.

6.2 Lancement

Le service s'appelle **named**.

```
| # service named start
```

Ou :

```
| # /etc/init.d/named start
```

6.3 Configuration de Bind

Bind (*Berkeley Internet Name Daemon*) est le serveur de noms le plus utilisé sur Internet. Bind 9 supporte l'IPv6, les noms de domaine unicode, le multithread et de nombreuses améliorations de sécurité.

6.3.1 Configuration générale

La configuration globale de Bind est placée dans le fichier `/etc/named.conf`. La configuration détaillée des zones est placée dans `/var/lib/named`. `/etc/named.conf` est composé de deux parties. La première concerne la configuration globale des options de Bind. La seconde est la déclaration des zones pour les domaines individuels. Les commentaires commencent par un `#` ou `//`.

■ Remarque

Attention il arrive parfois (notamment sur RHEL 4.x) que la configuration de Bind soit « chrootée » (déplacée dans une arborescence spécifique d'où le service ne peut sortir, le reste de l'arborescence lui étant inaccessible). Sur Centos et RHEL 4.x et supérieurs named.conf est dans /var/named/chroot/etc/. On peut modifier ce mode en modifiant le fichier de configuration /etc/sysconfig/named.

```
# cat /etc/sysconfig/named
...
CHROOT=/var/named/chroot
...
```

Dans ce cas, tous les fichiers de configuration, y compris les zones, sont relatifs à ce chemin. Voici un fichier named.conf de base.

```
options {
    directory "/var/lib/named";
    forwarders { 10.0.0.1; };
    notify no;
};

zone "localhost" in {
    type master;
    file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
    type master;
    file "127.0.0.zone";
};

zone "." in {
    type hint;
    file "root.hint";
};
```

6.3.2 Section globale

La configuration globale est placée dans la section **options**. Voici un détail de quelques options importantes (le point-virgule doit être précisé) :

- **directory "filename";** : emplacement des fichiers contenant les données des zones.
- **forwarders { adresse-ip; };** : si le serveur bind ne peut résoudre lui-même la requête, elle est renvoyée à un serveur DNS extérieur, par exemple celui du fournisseur d'accès.
- **listen-on-port 53 {127.0.0.1; adresse-ip; };** : port d'écoute du DNS suivi des adresses d'écoute. On indique ici les adresses IP des interfaces réseau de la machine. Il ne faut pas oublier 127.0.0.1.

- **allow-query { 127.0.0.1; réseau; };** : machine(s) ou réseau(x) autorisé(s) à utiliser le service DNS. Par exemple 192.168.1/24. Si la directive est absente, tout est autorisé.
- **allow-transfer { 192.168.1.2; };** : machine(s) ou réseau(x) autorisé(s) à copier la base de données dans le cas d'une relation maître et esclave. Par défaut aucune copie n'est autorisée.
- **notify no;** : on notifie ou non les autres serveurs DNS d'un changement dans les zones ou d'un redémarrage du serveur.

6.3.3 Section de zones

Pour chaque domaine ou sous-domaine, on définit deux sections **zone**. La première contient les informations de résolution de nom (nom vers IP) et la seconde les informations de résolution inverse (IP vers Nom). Dans chacun des cas, la zone peut être maître **Master** ou esclave **Slave** :

- **Master** : le serveur contient la totalité des enregistrements de la zone dans ses fichiers de zone. Lorsqu'il reçoit une requête, il cherche dans ses fichiers (ou dans son cache) la résolution de celle-ci.
- **Slave** : le serveur ne contient par défaut aucun enregistrement. Il se synchronise avec un serveur maître duquel il récupère toutes les informations de zone. Ces informations peuvent être placées dans un fichier. Dans ce cas l'esclave stocke une copie locale de la base. Lors de la synchronisation, le numéro de série de cette copie est comparé à celui du maître. Si les numéros sont différents, une nouvelle copie a lieu, sinon la précédente continue à être utilisée.

6.3.4 Zone de résolution

Elle est généralement appelée **zone**. Pour chaque domaine ou sous-domaine, elle indique dans quel fichier sont placées les informations de la zone (c'est-à-dire entre autres les adresses IP associées à chaque hôte), son type (maître ou esclave), si on autorise ou non la notification, l'adresse IP du serveur DNS maître dans le cas d'un esclave, etc.

Le nom de la zone est très important puisque c'est lui qui détermine le domaine de recherche. Quand le DNS reçoit une requête, il recherche dans toutes les zones une correspondance.

```
zone "domaine.org" {
    type      "master";
    file     "domaine.org.zone";
};
```

- **type** : master ou slave.

- **file** : nom du fichier qui contient les informations de la zone. Il n'y a pas de règles précises de nommage mais pour des raisons de lisibilité il est conseillé de lui donner le même nom que la zone tant pour une zone master que pour une slave. Pour un master, c'est l'original éventuellement rempli par vos soins. Pour un slave, ce n'est pas obligatoire. S'il est présent, ce sera une copie du master, synchronisée.
- Dans le cas d'un Master, on peut rajouter **allow-transfer** (serveurs autorisés à dupliquer la zone) et **notify yes** (indique une mise à jour ou une relance pour les slaves).

En cas de Slave : on rajoute la directive **masters** pour indiquer à partir de quel serveur Master dupliquer.

6.3.5 Zone de résolution inverse

Pour chaque réseau ou sous-réseau IP (ou plage d'adresses) on définit une zone de résolution inverse dont le fichier contient une association IP vers nom de machine. C'est en fait presque la même chose que la zone de résolution sauf que l'on doit respecter une convention de nommage :

- Le nom de la zone se termine toujours par une domaine spécial **.in-addr.arpa**.
- On doit tout d'abord déterminer quel réseau la zone doit couvrir (cas des sous-réseaux). Pour nous : un réseau de classe C 192.168.1.0 soit **192.168.1/24**.
- On inverse l'ordre des octets dans l'adresse : **1.168.192**.
- On ajoute **in-addr.arpa**. Notre nom de zone sera donc **1.168.192.in-addr.arpa**.
- Pour le reste, les mêmes remarques que pour la zone de résolution s'appliquent.

```
Zone "1.168.192.in-addr.arpa" {
    type      master;
    file     "192.168.1.zone";
};
```

6.3.6 Exemple

Soit un domaine **domaine.org** sur un réseau de classe C 192.168.1.0. Soit deux serveurs DNS 192.168.1.1 Master et 192.168.1.2 Slave.

Sur le Master

```
zone "domaine.org" {
    type      master;
    file     "domaine.org.zone";
```

```

    allow-transfer { 192.168.1.2; } ;
    notify yes;
};

zone "1.168.192.in-addr.arpa" {
    type    master;
    file    "192.168.1.zone";
    allow-transfer { 192.168.1.2; } ;
    notify yes;
};

```

Sur le Slave

```

zone "domaine.org" {
    type      slave;
    file      "domaine.org.zone";
    masters   { 192.168.1.1; };

};

zone "1.168.192.in-addr.arpa" {
    type      slave;
    file      "192.168.1.zone";
    masters   { 192.168.1.1; };
};

```

6.3.7 Zones spéciales

La zone racine « . » permet de spécifier les serveurs racines. Quand aucune des zones n'arrive à résoudre une requête, c'est la zone racine qui est utilisée par défaut et qui renvoie sur les serveurs racines.

La zone de loopback n'est pas nécessaire bien que utile. Elle fait office de **cache DNS**. Quand une requête arrive sur le serveur et qu'il ne possède pas l'information de résolution, il va la demander aux serveurs DNS racines qui redescendront l'information. Celle-ci est alors placée en cache. Du coup les accès suivants seront bien plus rapides !

6.4 Fichiers de zones

6.4.1 Définitions

Les fichiers de zones utilisent plusieurs termes, caractères et abréviations spécifiques.

- **RR** : *Resource Record*. Nom d'un enregistrement DNS (les données du DNS).
- **SOA** : *Start Of Authority*. Permet de décrire la zone.
- **IN** : *the Internet*. Définit une classe d'enregistrement qui correspond aux données Internet (IP). C'est celle par défaut si elle n'est pas précisée pour les enregistrements.

- **A** : *Address*. Permet d'associer une adresse IP à un nom d'hôte. Pour IPv6 c'est AAAA.
- **NS** : *Name Server*. Désigne un serveur DNS de la zone.
- **MX** : *Mail eXchanger*. Désigne un serveur de courrier électronique, avec un indicateur de priorité. Plus la valeur est faible, plus la priorité est élevée.
- **CNAME** : *Canonical Name*. Permet de rajouter des alias : lier un nom à un autre. On peut créer des alias sur des noms d'hôte et aussi sur des alias.
- **PTR** : *Pointer*. Dans une zone de résolution inverse, fait pointer une IP sur un nom d'hôte.
- **TTL** : *Time To Live*. Durée de vie des enregistrements de la zone.
- **@** : dans les déclarations de la zone, c'est un alias (caractère de remplacement) pour le nom de la zone déclarée dans /etc/named.conf. Ainsi si la zone s'appelle domaine.org, @ vaut domaine.org. Dans la déclaration de l'administrateur de la SOA, il remplace ponctuellement le point dans l'adresse de courrier électronique.
- Le point « . »: Si l'on omet le point en fin de déclaration d'hôte, le nom de la zone est concaténé à la fin du nom. Par exemple pour la zone domaine.org, si on écrit **poste1**, cela équivaut à **poste1.domaine.org**. Si on écrit **poste1.domaine.org** (sans le point à la fin) alors on obtient comme résultat **post1.domaine.org.domaine.org** ! Pour éviter cela, vous devez écrire **poste1.domaine.org**. (notez le point à la fin).
- Certains enregistrements nécessitent une notion de durée, qui est généralement exprimée en secondes, mais aussi parfois avec des abréviations :
 - **1M** : une minute, soit 60 secondes (1M, 10M, 30M, etc.).
 - **1H** : une heure, 3600 secondes.
 - **1D** : un jour, 86400 secondes.
 - **1W** : une semaine, 604800 secondes.
 - **365D** : un an, 31536000 secondes.

■ Remarque

Attention, et ceci est très important : dans les fichiers de zones, IL NE FAUT JAMAIS COMMENCER UNE LIGNE PAR DES ESPACES OU TABULATIONS. Ça ne marche absolument pas : les espaces ou tabulations seraient interprétés comme faisant partie du nom indiqué, de l'adresse ou de l'option.

6.4.2 Zone

Commencez tout d'abord par une directive **TTL** qui indique le temps de vie de la zone en secondes. Cela signifie que chaque enregistrement de la zone sera valable durant le temps indiqué par **TTL** (note : il est possible de modifiter cette valeur pour chaque enregistrement). Durant ce temps, les données peuvent être placées

en cache par les autres serveurs de noms distants. Une valeur élevée permet de réduire le nombre de requêtes effectuées et de rallonger les délais entre les synchronisations.

```
| $TTL 86400
```

Après les directives TTL, placez un enregistrement de ressources **SOA** :

```
| <domain>      IN      SOA      <primary-name-server> <hostmaster-email> (  
|   <serial-number>  
|   <time-to-refresh>  
|   <time-to-retry>  
|   <time-to-expire>  
|   <minimum-TTL> )
```

- **domain** : c'est le nom de la zone, le même nom que celui utilisé dans /etc/named.conf. On peut le remplacer par @ sinon il ne faut pas oublier de le terminer par un point (pour éviter une concaténation).
- **primary-name-server** : le nom sur le serveur DNS maître sur cette zone. Il ne faudra pas oublier de le déclarer dans la liste des hôtes (enregistrements PTR ou A).
- **hostmaster-email** : adresse de courrier électronique de l'administrateur du serveur de nom. Le caractère @ étant déjà réservé à un autre usage, on utilise un point pour le remplacer. Ainsi « admin@domaine.org » devra s'écrire « **admin.domaine.org.** » .
- **serial-number** : c'est un numéro de série que l'on doit incrémenter manuellement à chaque modification du fichier zone pour que le serveur de nom sache qu'il doit recharger cette zone. Elle est utilisée pour la synchronisation avec les serveurs esclaves. Si le numéro de série est le même qu'à la dernière synchronisation les données ne sont pas rafraîchies. Par convention on place **YYYYMMDDNN** (année-mois-jour-numéro) sur dix chiffres.
- **time-to-refresh** : indique à tout serveur esclave combien de temps il doit attendre avant de demander au serveur de noms maître si des changements ont été effectués dans la zone.
- **time-to-retry** : indique au serveur esclave combien de temps attendre avant d'émettre à nouveau une demande de rafraîchissement si le serveur maître n'a pas répondu. La demande aura lieu toutes les time-to-retry secondes.
- **time-to-expire** : si malgré les tentatives de contacts toutes les time-to-retry secondes le serveur n'a pas répondu au bout de la durée indiquée dans time-to-expire, le serveur esclave cesse de répondre aux requêtes pour cette zone.

- **Minimum-TTL** : le serveur de nom demande aux autres serveurs de noms de mettre en cache les informations pour cette zone pendant au moins la durée indiquée.

```
| @ IN SOA dns1.domaine.org. hostmaster.domaine.org. (  
| 2005122701 ; serial  
| 21600 ; refresh de 6 heures  
| 3600 ; tenter toutes les 1 heures  
| 604800 ; tentatives expirer après une semaine  
| 86400 ) ; TTL mini d'un jour
```

Passez ensuite aux enregistrements **NS** (*Name Server*) où vous spécifiez les serveurs de noms de cette zone.

```
| IN NS dns1  
| IN NS dns2
```

■ Remarque

Quand on ne spécifie pas en début de ligne un nom d'hôte ou de zone (complet ou @), cela veut dire qu'on utilise le même que la ligne du dessus. Tant qu'on n'en précise pas de nouveau, c'est le dernier indiqué qui est utilisé. Ainsi ci-dessus les lignes pourraient être :

```
| @ IN NS dns1  
| @ IN NS dns2
```

ou :

```
| domaine.org. IN NS dns1  
| domaine.org. IN NS dns2
```

■ Remarque

Notez l'absence de point après le nom de l'hôte et donc domaine.org est concaténé pour obtenir dns1.domaine.org.

```
| IN NS dns1
```

équivaut à :

```
| IN NS dns1.domaine.org.
```

Passez ensuite à l'énumération des serveurs de courrier électronique de la zone. La valeur numérique située après MX indique la priorité. Plus la valeur est basse plus le serveur est prioritaire et susceptible d'être contacté en premier. Si les valeurs sont identiques, le courrier est redistribué de manière homogène entre les serveurs.

Si un serveur ne répond pas (chargé, en panne) la bascule vers une autre machine est automatique.

```
| IN      MX      10      mail  
| IN      MX      15      mail2
```

Si vous souhaitez qu'une machine réponde en passant par le FQDN domaine.org sans préciser d'hôte (par exemple `http://domaine.org` sans utiliser `http://www.domaine.org`) alors vous pouvez maintenant déclarer une adresse IP pour ce serveur. Ainsi la commande `ping domaine.org` répondra 192.168.1.3 !

```
| IN A 192.168.1.3
```

Vous pouvez maintenant déclarer les autres hôtes dont les serveurs de noms, de mails, les postes, etc.

```
| dns1      IN      A      192.168.1.1  
| dns2      IN      A      192.168.1.2  
| server1   IN      A      192.168.1.3  
| server2   IN      A      192.168.1.4  
| poste1    IN      A      192.168.1.11  
| poste2    IN      A      192.168.1.12  
| poste3    IN      A      192.168.1.13
```

On remarque que nos serveurs mail et mail2 ne sont pas déclarés, et que l'on n'a pas indiqué de serveur Web et ftp. Nous allons utiliser les alias, en faisant pointer ces noms d'hôtes sur d'autres hôtes.

```
| mail      IN      CNAME    server1  
| mail2    IN      CNAME    server2  
| www      IN      CNAME    server1  
| ftp      IN      CNAME    server1
```

La configuration de la zone est terminée, il faut maintenant s'occuper de la zone de résolution inverse.

6.4.3 Zone de résolution inverse

La zone de résolution inverse est presque identique à la précédente, si ce n'est que les enregistrements A sont remplacés par des enregistrements PTR destinés à traduire une IP en hôte. Le TTL et la déclaration SOA doivent être si possible identiques (sauf le nom de la zone). Vous placez aussi les enregistrements NS.

```
| IN      NS      dns1.domaine.org.  
| IN      NS      dns2.domaine.org.
```

Vous n'êtes pas obligé de placer dans la zone de résolution inverse la traduction des adresses IP du DNS, étant donné que c'est le DNS lui-même qui résout son propre nom ! Cependant le faire peut accélérer la démarche, le DNS n'ayant pas à exécuter une requête sur lui-même. Passez aux enregistrements PTR traduisant l'adresse IP pour chaque hôte.

```
1      IN    PTR    dns1.domaine.org.  
2      IN    PTR    dns2.domaine.org.  
3      IN    PTR    server1.domaine.org.  
4      IN    PTR    server2.domaine.org.  
11     IN    PTR    poste1.domaine.org.  
12     IN    PTR    poste2.domaine.org.  
13     IN    PTR    poste3.domaine.org.
```

Il est théoriquement possible pour la même IP d'attribuer plusieurs hôtes ; les RFC ne sont pas très explicites sur cette possibilité qui, au final, peut créer des problèmes.

6.5 Diagnostic des problèmes de configuration

La commande **named-checkconf** vérifie la syntaxe du fichier **named.conf**. Vous lui fournissez en paramètre le fichier. La sortie indiquera les lignes posant problème.

La commande **named-checkzone** vérifie la syntaxe d'un fichier de zone (y compris de résolution inverse). Vous lui spécifiez en paramètre le nom du fichier zone.

6.5.1 Interrogation dig et host

Le programme **dig** est un outil d'interrogation avancé de serveur de noms, capable de restituer toutes les informations des zones.

```
> dig free.fr  
  
; <>> DiG 9.4.1-P1 <>> free.fr  
;; global options: printcmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63972  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0  
  
;; QUESTION SECTION:  
;free.fr.                      IN      A  
  
;; ANSWER SECTION:  
free.fr.                      86363   IN      A      212.27.48.10  
  
;; Query time: 1 msec  
;; SERVER: 10.23.254.240#53(10.23.254.240)
```

```
;; WHEN: Wed May 14 09:36:09 2008
;; MSG SIZE  rcvd: 41
```

Par défaut dig ne restitue que l'adresse de l'hôte passé en paramètre. En cas de réussite, le statut vaut **NOERROR**, le nombre de réponses est indiqué par **ANSWER** et la réponse se situe en dessous de la section **ANSWER**. Pour obtenir une résolution inverse il existe deux solutions.

```
■ $ dig 10.48.27.212.in-addr.arpa ptr
```

ou plus simplement :

```
$ dig -x 212.27.48.10

; <>> DiG 9.4.1-P1 <>> -x 212.27.48.10
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60222
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;10.48.27.212.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
10.48.27.212.in-addr.arpa. 86400 IN      PTR      www.free.fr.

;; Query time: 31 msec
;; SERVER: 10.23.254.240#53(10.23.254.240)
;; WHEN: Wed May 14 09:36:51 2008
;; MSG SIZE  rcvd: 68
```

Dans la première syntaxe, remarquez que vous pouvez rajouter un paramètre d'interrogation. Voici les principaux.

- **a** : uniquement l'adresse.
- **any** : toutes les informations concernant le domaine.
- **mx** : les serveurs de messagerie.
- **ns** : les serveurs de noms.
- **soa** : la zone Start of Authority.
- **hinfo** : infos sur l'hôte.
- **txt** : texte de description.
- **ptr** : zone reverse de l'hôte.
- **axfr** : liste de tous les hôtes de la zone.

```
■ $ dig free.fr any
; <>> DiG 9.4.1-P1 <>> free.fr any
;; global options:  printcmd
```

```
; Got answer:  
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28893  
; flags: qr aa; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 8  
  
; QUESTION SECTION:  
;free.fr. IN ANY  
  
; ANSWER SECTION:  
free.fr. 86400 IN NS freens2-g20.free.fr.  
free.fr. 86400 IN A 212.27.48.10  
free.fr. 86400 IN NS freens1-g20.free.fr.  
free.fr. 86400 IN MX 20 mx2.free.fr.  
free.fr. 86400 IN SOA freens1-g20.free.fr.  
hostmaster.proxad.net. 2008051001 10800 3600 604800 86400  
free.fr. 86400 IN MX 10 mx1.free.fr.  
  
; ADDITIONAL SECTION:  
freens2-g20.free.fr. 86400 IN A 212.27.60.20  
mx1.free.fr. 86400 IN A 212.27.48.6  
mx2.free.fr. 86400 IN A 212.27.42.56  
freens1-g20.free.fr. 86400 IN A 212.27.60.19  
mx2.free.fr. 86400 IN A 212.27.42.58  
mx1.free.fr. 86400 IN A 212.27.48.7  
mx2.free.fr. 86400 IN A 212.27.42.57  
mx2.free.fr. 86400 IN A 212.27.42.59  
  
; Query time: 9 msec  
; SERVER: 10.23.254.240#53(10.23.254.240)  
; WHEN: Wed May 14 09:35:32 2008  
; MSG SIZE rcvd: 318
```

L'outil **host** fournit le même résultat de manière peut-être un peu plus simple.

```
$ host free.fr  
free.fr has address 212.27.48.10  
free.fr mail is handled by 10 mx1.free.fr.  
  
$ host -t any free.fr  
free.fr has address 212.27.48.10  
free.fr name server freens1-g20.free.fr.  
free.fr has SOA record freens1-g20.free.fr, hostmaster.proxad.net.  
2008051001 10800 3600 604800 86400  
free.fr mail is handled by 10 mx1.free.fr.  
  
$ host -a free.fr  
Trying "free.fr"  
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64513  
; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 2  
  
; QUESTION SECTION:  
;free.fr. IN ANY
```

```
;; ANSWER SECTION:  
free.fr.          86140    IN      A       212.27.48.10  
free.fr.          86140    IN      NS      freensl-g20.free.fr.  
free.fr.          86140    IN      SOA     freensl-g20.free.fr.  
hostmaster.proxad.net. 2008051001 10800 3600 604800 86400  
free.fr.          86140    IN      MX      10 mx1.free.fr.  
  
;; ADDITIONAL SECTION:  
freensl-g20.free.fr. 86140    IN      A       212.27.60.19  
mx1.free.fr.        86140    IN      A       212.27.48.7  
  
Received 176 bytes from 10.23.254.240#53 in 4 ms
```

7. Courier électronique

7.1 Principe

- Quand un client (un utilisateur) envoie un message, il utilise un **MUA** (*Mail User Agent*), par exemple Outlook Express, Thunderbird, Evolution, Kmail, Mutt, etc.
- Le **MUA** envoie le message au **MTA** (*Mail Transport Agent*). Le MTA étudie l'adresse électronique pour isoler l'utilisateur et le domaine de destination. Puis il vérifie les informations DNS de type **MX** (*Mail exchanger*) pour le domaine choisi, pour savoir à quel serveur transmettre le courrier. Si aucun MTA n'est disponible, le message est placé en file d'attente et relance la distribution plus tard (le délai dépend de la configuration du MTA).
- Le MX peut être soit un autre MTA, qui jouera le rôle de routeur (cas d'une redirection vers un sous-domaine par exemple), soit un **MDA** (*Mail Delivery Agent*). Le MDA place le message dans un fichier temporaire, peut le filtrer, etc.
- À ce niveau, le destinataire reçoit le message : soit il le récupère en lisant directement le fichier temporaire (cas de la commande mail par exemple) soit il passe par un protocole de type **POP** ou **IMAP**.
- Le protocole de transport de messages est le **SMTP** (*Simple Mail Transfer Protocol*) sur le port 25.
- Les protocoles de réception de messages soit **POP** (*Post Office Protocol*) sur le port 110 (POP3), soit **IMAP** (*Internet Message Access Protocol*).

Deux suites de courrier électronique se partagent l'essentiel du marché sur Unix : **sendmail** et **postfix**.

La suite libre **sendmail** est la plus connue et la plus utilisée. Sendmail a été créé en 1981 par Eric Allman et a été intégré à BSD 4.2 en 1983. On estimait en 2000 son utilisation à plus de 100 millions de serveurs de courrier électronique. Tant qu'il ne faut pas modifier fortement sa configuration de base, sendmail est idéal.