Mise en place d'un réseau sous Linux

Julien De Bona, julien@bawet.org

Résumé

Ce document explique comment configurer des machines Linux dans un réseau local (LAN) connecté à l'Internet et y installer divers services. Ce document n'est pas autonome : il sert de support à un petit cours sur les réseaux sous Linux.

Première partie

Configuration du réseau

1 Description du réseau mis en place

Les exemples de ce document ont été réalisés sur un réseau formé de deux machines.

1.1 Le serveur

La première machine, tournant sous Debian GNU/Linux 3.0 (Woody) avec un kernel 2.4, sera appelée ***julien***. Comme elle hébergera divers services, elle sera aussi appelée "le serveur". Elle possède une carte Ethernet PCI de type ne2000 connectée à un modem câble (connexion à Internet via Télédisnet) et une interface tuntap[1] qui la connectera au réseau local.

1.2 Le client

La seconde machine,tournant sous Debian GNU/Linux 2.2 (Potato), sera appelée ***tux*** et utilisera les services proposés par le serveur ; elle sera donc aussi appelée "le client". Elle possède une carte réseau ISA de type ne2000 connectée au réseau local.
tux est en fait une machine virtuelle émulée grâce au programme Bochs[2], et le réseau local est donc lui aussi virtuel. Si ***tux*** avait été un véritable ordinateur, ***julien*** aurait possédé deux cartes Ethernet.

[1] une interface tuntap est en quelque sorte une interface Ethernet virtuelle

[2] http://bochs.sourceforge.net

2 Chargement des pilotes des cartes réseaux

> Configuration requise:
> un kernel compilé avec les pilotes des cartes réseaux utilisées et le package modutils (c'est le cas de toutes les distributions actuelles. Pour ceux qui voudraient compiler eux-mêmes leur kernel, il est plus flexible de compiler les pilotes en modules.

Sous Linux, les interfaces Ethernet sont désignées par eth0, eth1... dans l'ordre de leur détection. Les interfaces virtuelle tuntap de ***julien***, elles, sont désignées par tun0, tun1...
Si les cartes réseaux ont été configurées lors de l'installation, vous pouvez passer cette section.
Les pilotes sont disponibles de deux manières : soit ils sont inclus dans le kernel, auquel cas il n'y a pas besoin de le charger (c'est de plus en plus rare), soit ils sont contenus dans des modules que l'on charge avec la commande modprobe(8) ou insmod(8). Le chargement des modules peut se faire à chaque démarrage en ajoutant son nom dans /etc/modules(5) (propre à Debian) ou lorsqu'on en a besoin en mettant dans /etc/modules.conf(5) [3] une ligne contenant le nom du périphérique (ex : eth0 pour la 1ère carte Ethernet ou char-major-10-200[4] pour les interfaces tuntap) et le nom du module correspondant.
Pour savoir quel module utiliser, consulter l'ETHERNET-HOWTO ou les documentations concernant le kernel (dans /usr/src/linux/Documentation).

2.1 le serveur

Sur le serveur, le pilote de la carte Ethernet s'appelle ne2k-pci et est inclus dans le kernel : il n'y a rien à faire. Le pilote de l'interface tuntap, lui, est déjà mentionné dans /etc/modules.conf(5) et est donc déjà chargé à la demande :
/etc/modules.conf sur ***julien*** (extrait)

```
### This file is automatically generated by update-modules"
#
# Please do not edit this file directly. If you want to change or add
# anything please take a look at the files in /etc/modutils and read
# the manpage for update-modules.
#
alias char-major-10-200        tun
```

[3] !!! Sous Debian, il ne faut pas manipuler ce fichier directement : il faut éditer les fichiers contenus dans le répertoire /etc/modutils puis lancer la commande update-modules(8), qui créera le fichier /etc/modules.conf(5) à partir de ces fichiers.

[4] Pour savoir à quoi correspondent les notations char-major-x-y, consulter le fichier Documentation/devices.txt dans les sources du kernel (/usr/src/linux).

2.2 le client

Sur le client, le pilote de la carte Ethernet s'appelle ne, et on charge ce pilote au démarrage : /etc/modules sur ***tux***

```
# /etc/modules: kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line.  Comments begin with
# a "#", and everything on the line after them are ignored.

ne
```

2.3 Le cas d'une machine avec plusieurs cartes réseaux

Le cas le plus simple est celui de cartes utilisant des pilotes différents compilés en modules : il suffit de charger l'un après l'autre les deux modules, et chacun reconnaîtra "sa" carte. L'ordre du chargement des modules détermine le nom qui sera attribué aux interfaces réseaux : eth0, eth1 ...

Si un des pilotes est compilé dans le kernel, alors la carte qui lui correspond sera détectée au démarrage, et il restera à charger le module pour l'autre carte.

Si les deux pilotes sont compilés dans le kernel, il faut passer au kernel lors du démarrage l'emplacement de chaque carte (irq, plage d'I/O), par exemple grâce à une ligne "append=..." dans /etc/lilo.conf(5). En effet, le kernel ne prend la peine de ne détecter qu'une seule carte par défaut. Pour chaque carte, la syntaxe est

`ether=irq,plage_io,[autres_paramètres,]nom`

, ce qui donne par exemple

`append="ether=5,0x340,eth0 ether=9,0x300,eth1"`

.

Le cas le plus compliqué est celui de deux cartes ayant le même pilote. Si les pilotes sont compilés dans le kernel, la procédure est identique. Si on a affaire à un module, il faut le charger deux fois avec à chaque fois en passant sur la ligne de commande les paramètres de chaque page sous la forme

`paramètre=valeur`

. La liste des paramètres peut être obtenue grâce à la commande modinfo(8), à laquelle on passe en argument le nom du module.

Cela donne par exemple

`modprobe ne irq=9 io=0x300`

.

3 Configuration de la couche IP

3.1 Configuration du réseau

> Configuration requise:
> le package netbase (installé par défaut sous Debian) et, si on veut se connecter à Internet, le package dhcp-client. Les packages pump et dhclient peuvent remplacer dhcp-client, mais ce premier offre plus de possibilités de configuration, qui seront utilisées plus loin dans ce document. D'autres packages peuvent venir s'ajouter pour une connexion Internet par modem classique ou ADSL.

Dans un réseau, chaque machine (ou plus précisément chaque interface réseau) a besoin de paramètres pour préciser à quel réseau elle appartient et à l'identifier sur ce réseau. ***julien*** est particulier parce qu'il est sur deux réseaux à la fois : son interface eth0 le connecte au réseau de Télédis, qui est lui-même connecté à Internet et son interface tun0 le connecte au réseau local.
Ces paramètres sont :

– l'adresse IP
– le masque de sous-réseau
– l'adresse du réseau
– l'adresse de broadcast

En fait, l'adresse du réseau et l'adresse de broadcast dépendent de l'adresse IP et du masque de sous-réseau, et il n'est donc généralement pas nécessaire de les spécifier lors de la configuration. L'adresse du réseau s'obtient en faisant un "et" binaire entre l'adresse IP et le masque, tandis que l'adresse de broadcast est le résultat du "ou" binaire entre l'adresse IP et le "non" binaire du masque. Pour créer un réseau, il faut assigner aux ordinateurs le même masque de sous-réseau et une adresse IP distincte telle que tous les ordinateurs du réseau aient la même adresse de réseau. Les adresses de réseau et de broadcast ne peuvent pas être allouées à un ordinateur.
Un exemple de paramètres :

Adresse IP (décimal)	`192.168. 1. 1`
Masque de sous-réseau	`255.255.255. 0`
Adresse IP (binaire)	`11000000.10101000.00000001.00000001`
Masque de sous-réseau (binaire)	`11111111.11111111.11111111.00000000`
Non Masque de sous-réseau	`00000000.00000000.00000000.11111111`
Réseau = adresse IP et masque de sous-réseau (binaire)	`11000000.10101000.00000001.00000000`
Broadcast = adresse IP ou non masque (binaire)	`11000000.10101000.00000001.11111111`
Réseau (décimal)	`192.168.1.0`
Broadcast (décimal)	`192.168.1.255`

Pour rappel sur les opérateurs binaires, voici le "et" et le "ou"

a	b	a et b	a ou b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

et la négation

a	non a
0	1
1	0

Pour l'exemple, on utilisera le masque de sous-réseau `255.255.255.0`. Les adresses IP, elles, seront choisies dans la plage de `192.168.1.1` à `192.168.1.254`. ***julien*** va recevoir l'adresse `192.168.1.1` et ***tux*** l'adresse `192.168.1.2`.
Ces paramètres sont fréquemment utilisés dans le cadre de réseaux locaux, car cette plage d'adresses est réservée à cet usage et ne causera aucun problème quand le réseau sera connecté à Internet. Des adresses de réseau du type `192.168.x.0` (avec le masque `255.255.255.0`) peuvent également être utilisées si on désire créer plusieurs réseaux.
Cette configuration se fait avec la commande ifconfig(8). Cela donne
`ifconfig tun0 192.168.1.1 netmask 255.255.255.0`
sur ***julien*** et
`ifconfig eth0 192.168.1.2 netmask 255.255.255.0`
sur ***tux***. Sur Debian, cette configuration est enregistrée dans le fichier /etc/network/interfaces(5). Sur ***julien***, il contient ceci :
/etc/network/interfaces sur ***julien***

```
# /etc/network/interfaces - configuration file for ifup(8), ifdown(8)

# The loopback interface
# Cette interface est obligatoire et configurée par l'installation.
# Elle est utilisée en interne par le système et permet de simuler un réseau
# même sans avoir accès à un réseau par une carte Ethernet, un modem, ...
# Une ligne "auto" indique que les interfaces qui suivent doivent être
# activées au démarrage
auto lo
iface lo inet loopback

# Teledisnet (configuration d'une interface par DHCP)
iface eth0 inet dhcp

# Pour le réseau local (configuration d'une interface en fixant soi-même les
# paramètres).
iface tun0 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
```

.
On remarque que eth0, l'interface connectée à Internet, n'est pas activée au démarrage : elle le sera après que le firewall ait été mis en place pour des raisons de sécurité. L'interface tun0 n'est

pas activée au démarrage car elle n'existe que quand la machine virtuelle Bochs fonctionne et est activée à ce moment là. S'il s'agit d'une vraie interface Ethernet, elle peut franchement être activée par défaut (à moins qu'on ne veuille se protéger du réseau local avec le firewall).
Sur ***tux***, il contient ceci :
/etc/network/interfaces sur ***tux***

```
# /etc/network/interfaces - configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# Pour le réseau local
# La ligne gateway désigne l'ordinateur du réseau qui est connecté à Internet
# Ce ne sera utile que pour le partage de connexion.
auto eth0
iface eth0 inet static
        address 192.168.1.2
        netmask 255.255.255.0
        network 192.168.1.0
        gateway 192.168.1.1
        broadcast 192.168.1.255
```

On utilise alors les commandes ifup(8) et ifdown(8) avec en argument le nom de l'interface pour l'activer ou la désactiver.
Pour tester la configuration du réseau, on utilise la commande ping(8). Sur ***julien***, taper
`ping -c 5 192.168.1.2`
, et sur ***tux***,
`ping -c 5 192.168.1.1`
. Si la sortie ressemble à

```
PING 192.168.1.2 (192.168.1.2) : 56 data bytes
64 bytes from 192.168.1.2 : icmp_seq=0 ttl=255 time=5.7 ms
64 bytes from 192.168.1.2 : icmp_seq=0 ttl=255 time=5.7 ms
64 bytes from 192.168.1.2 : icmp_seq=1 ttl=255 time=2.0 ms
64 bytes from 192.168.1.2 : icmp_seq=2 ttl=255 time=2.1 ms
64 bytes from 192.168.1.2 : icmp_seq=3 ttl=255 time=2.1 ms
```

c'est que tout va bien. Sinon, vérifier le câblage et la configuration.

3.2 Configuration du firewall et partage de la connexion Internet

Configuration requise:
julien, qui partagera la connexion, a besoin d'un kernel 2.4 configuré avec netfilter (par défaut dans la plupart des distributions actuelles (Debian utilise par défaut un kernel 2.2, mais propose aussi un kernel 2.4 à l'installation (2.4.18-bf24) contenant netfilter)). Le package iptables permettra d'accéder à ces fonctionnalités. ***tux*** n'a pas besoin de packages supplémentaires. Il est possible de faire un firewall avec un kernel 2.2 grâce à ipchains, mais les règles sont quelque peu différentes.

3.2.1 Le fonctionnement de netfilter et iptables

NB: Netfilter est le nom du firewall, tandis qu'iptables est le nom de la commande permettant de le configurer
Netfilter est un dispositif de filtrage du trafic IP. Chaque paquet IP passant par l'ordinateur est examiné et des traitements lui sont appliqués. Netfilter possède des "tables" qui permettent chacune l'exécution de certains traitements. Les deux tables les plus importantes sont "filter", qui permet de bloquer ou d'accepter des paquets et "nat", qui permet de changer les adresses d'origine ou de destination du paquet et de ceux qui suivront. Chaque table possède des "chaînes" à travers lesquelles passent les paquets et qui spécifient les traitements à leur appliquer. Quand un paquet passe dans une chaîne, on regarde s'il correspond à la 1ère règle de la chaîne. S'il concorde, le traitement indiqué par la règle est appliqué et le parcours de la chaîne est interrompu. Sinon, l'opération est répétée avec la règle suivante. Si le paquet ne correspond à aucune règle, la politique par défaut de la chaîne est appliquée.
Les principaux usages d'iptables(8) sont :

```
iptables -t table -P chaîne politique
```
(spécifie une politique par défaut pour une chaîne)

```
iptables -t table -A chaîne critère -j action
```
(ajoute à la fin d'une chaîne une nouvelle règle, indiquant l'action à réaliser si le paquet vérifie le critère)

```
iptables -t table -F chaîne
```
(efface toutes les règles de la chaîne)

La table filter Cette table sert principalement à accepter ou rejeter des paquets, et compte trois chaînes :
- INPUT, à travers laquelle passent les paquets à destination de l'ordinateur
- OUTPUT, à travers laquelle passent les paquets provenant de l'ordinateur
- FORWARD, à travers laquelle passent les paquets passant par l'ordinateur (par exemple, du réseau local vers Internet et vice-versa) ; ces paquets ne passent pas par les chaînes INPUT et OUTPUT

Les actions (ce qu'on met après l'option -j) qui y sont possibles sont
– ACCEPT (accepter le paquet)
– DROP (rejeter le paquet, faire comme s'il n'était jamais arrivé)
– REJECT (rejeter le paquet, et envoyer à son expéditeur un message)
– LOG (logger le paquet par l'intermédiaire de syslogd ; on retrouvera des traces dans /var/log ; contrairement à la règle, les paquets loggés poursuivent leur chemin dans la chaîne)
Cette table est utilisée par défaut si on ne donne pas l'option -t à iptables.

La table nat Cette table sert à modifier les adresses d'origine. Seuls les paquets initiant une connexion passent par cette table ; ceux qui suivent subissent automatiquement le même traitement. Elle possède aussi trois chaînes :
– OUTPUT, par laquelle passent les paquets générés par l'ordinateur
– PREROUTING, qui traite les paquets avant d'examiner par où ils doivent être dirigés
– POSTROUTING, qui traite les paquets juste avant qu'ils ne partent pour leur destination
Les actions possibles les plus courantes sont
– MASQUERADE (accessible seulement dans POSTROUTING, change l'adresse source en l'adresse de l'ordinateur ; les réponses à ce paquet seront automatiquement redirigées vers la vraie adresse source ; l'application de cette règle est le partage de connexion
– DNAT (accessible seulement dans PREROUTING et OUTPUT, modifie la destination du paquet, indiquée par –to-destination adresse_ip. C'est utile si on a un serveur dans le réseau local qu'on veut rendre accessible de l'extérieur.

Les règles de concordance des paquets Ces options sont à passer à iptables pour décrire les paquets à traiter

-p protocole (protocole=tcp, udp ou icmp) : tcp est utilisé pour la plupart des applications (http, ftp, . . .), udp principalement par le dns et icmp par le ping

–dport port indique le port de destination, si on a spécifié le protocole tcp ou udp (http = 80, ftp = 21, ssh = 22, telnet = 23), consulter /etc/services(5) pour en avoir la liste complète.

–sport port fonctionne de même, mais avec le port d'origine

-i interface (interface=eth0, eth1, tun0, . . .) indique l'interface par laquelle le paquet est arrivé à l'ordinateur

-o interface indique l'interface par laquelle le paquet va sortir

-s source indique l'adresse IP d'origine du paquet. Une plage d'adresses peut être spécifiée grâce à la notation adresse_ip/masque.

-d destination fonctionne de même avec l'adresse de destination du paquet

-m state –state état indique l'état de la connexion à laquelle le paquet appartient. Les valeurs possibles pour état sont
  – INVALID : le paquet ne correspond à aucune connexion
  – ESTABLISHED : le paquet appartient à une connexion pour laquelle des paquets ont circulé dans les deux sens

– NEW : le paquet commence une nouvelle connexion ou appartient à une connexion pour laquelle des paquets n'ont pas encore circulé dans les deux senso
– RELATED : le paquet commence une nouvelle connexion, mais cette connexion est associée à une autre (exemple : un transfert de données par ftp)

3.2.2 Le serveur

La commande iptables(8) permet de spécifier les traitements à appliquer aux données passant par l'ordinateur, comme par exemple les bloquer ou les rediriger. Le plus simple est de créer un script qui sera lancé à chaque démarrage.
La saine configuration d'un firewall implique de tout interdire par défaut, puis d'autoriser ce qui est nécessaire.
Tout d'abord, créer sur ***julien*** un fichier /etc/init.d/firewall contenant ceci :
/etc/init.d/firewall sur ***julien***

```
#!/bin/sh

PATH=/bin:/sbin:/usr/bin:/sbin:/usr/sbin:/usr/local/sbin
export PATH
COMMAND=/sbin/iptables
test -x $COMMAND || exit 0

# Définition des interfaces, à modifier pour correspondre à la configuration
# réelle
LOOPBACK=lo
INTERNET=eth0
LOCAL=tun0

# Si on ne sait pas quoi faire d'un paquet, on le bloque: c'est la bonnne
# manière de procéder pour avoir un firewall sécurisé.
INPUT_POLICY="DROP"
OUTPUT_POLICY="DROP"
FORWARD_POLICY="DROP"

case "$1" in
  start)
        echo -n "Configuration du firewall ($COMMAND)"
        # Les protocoles FTP et IRC sont plus complexes à gérer que les autres
        # Le chargement de ces modules permet leur traitement adéquat
        # par l'option -m d'iptables
        modprobe ip_conntrack_ftp
        modprobe ip_conntrack_irc
```

```
# Règles de filtrage pour INPUT
# Spécifier le comportement par défaut (bloquer)
$COMMAND -P INPUT $INPUT_POLICY
# Accepter tout ce qui vient dans l'interface loopback
$COMMAND -A INPUT -i "$LOOPBACK" -j ACCEPT
# Accepter tout ce qui vient d'Internet et qui appartient à une
# connexion établie (c'est-à-dire une donnée qu'on a demandée)
$COMMAND -A INPUT -i "$INTERNET" -m state -state RELATED,ESTABLISHED -j ACCEPT
# Accepter tout ce qui vient du réseau local
$COMMAND -A INPUT -i "$LOCAL" -j ACCEPT
# On aurait pu écrire:
# $COMMAND -A INPUT -s 192.168.1.1/255.255.255.0 -j ACCEPT


# Règles de filtrage pour les paquets sortants
$COMMAND -P OUTPUT $OUTPUT_POLICY
# Laisser passer tous les paquets sortants
$COMMAND -A OUTPUT -j ACCEPT

# Règles de filtrage pour FORWARD
$COMMAND -P FORWARD $FORWARD_POLICY
# Accepter ce qui vient du réseau local
$COMMAND -A FORWARD -i "$LOCAL" -j ACCEPT
# Accepter tout ce qui vient d'Internet et qui appartient à une
# connexion établie (c'est-à-dire une donnée qu'on a demandée)
$COMMAND -A FORWARD -i "$INTERNET" -m state -state RELATED,ESTABLISHED -j ACCEPT

# Activer le partage de connexion
$COMMAND -t nat -A POSTROUTING -o "$INTERNET" -j MASQUERADE
# Donner accès à un serveur SSH situé dans le réseau
$COMMAND -t nat -A PREROUTING -p tcp -dport ssh -j DNAT -to-destination 192.168.0.2
# Et autoriser cet accès
$COMMAND -A FORWARD -p tcp -dport ssh -j ACCEPT

# Se connecter à Internet (l'interface n'était pas activée
# automatiquement lors du démarrage (voir /etc/network/interfaces))
ifup $INTERNET

# Décommenter cette ligne si ip_forward n'est pas mis à yes dans
# /etc/network/options pour autoriser le passage des paquets
# entre internet et le réseau local.

# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
        ;;
  stop)
        echo "Déconfiguration du firewall ($COMMAND)"
        ifdown $INTERNET
        $COMMAND -P INPUT ACCEPT
        $COMMAND -P OUTPUT ACCEPT
        $COMMAND -P FORWARD ACCEPT
        $COMMAND -F
        $COMMAND -t nat -F
        echo "."
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  *)
        echo "Usage: /etc/init.d/firewall {start|stop|restart}"
        exit 1
esac

exit 0
```

.
Ce fichier doit appartenir à root et être exécutable par root : exécuter les commandes suivantes pour ce faire :
`chown root.root /etc/init.d/firewall ; chmod 755 /etc/init.d/firewall`

Ensuite, créer les liens nécessaires à son lancement à chaque démarrage par les commandes suivantes :
`cd /etc ; for i in rc[2-5].d ; do cd $i ; ln -s ../init.d/firewall S12firewall ;`
`cd .. ; done ; for i in rc[16].d ; do cd $i ; ln -s ../init.d/firewall K99firewall ;`
`cd .. ; done`

Il reste à autoriser le passage des données entre le réseau local et Internet. On peut soit ajouter la commande
`echo 1 > /proc/sys/net/ipv4/ip_forward`
au script dans la section "start", soit (propre à Debian), modifier le fichier /etc/network/options pour qu'il contienne la ligne suivante : `ip_forward=yes`.

3.2.3 Le client

tux, pour accéder à Internet, devra mentionner ***julien*** comme passerelle par défaut (gateway). Il doit pour cela lancer la commande
`route add default gw 192.168.1.1`

après l'activation de l'interface réseau. Le même effet est obtenu par la ligne `gateway 192.168.1.1` dans le fichier /etc/network/interfaces(5).

4 La résolution de noms

Configuration requise:
rien de particulier, c'est une fonctionnalité de la bibliothèque C (glibc, ou libc6

Il est plus simple de référencer un ordinateur par un nom du genre `www.bawet.org` que par son adresse `62.58.108.140`. Il y a deux manières pour cela :

– mentionner la correspondance nom-adresse dans le fichier /etc/hosts(5)
– mentionner un serveur DNS connaissant cette correspondance dans /etc/resolv.conf(5) ; pour gérer ainsi les machines du LAN, il faudra créer son propre serveur DNS (voir partie 2)

Dans le cas d'un réseau local avec peu de machines, on peut ajouter une ligne pour chaque ordinateur dans le fichier /etc/hosts(5) de chaque machine. L'inconvénient est qu'il faut le modifier sur chaque ordinateur à chaque fois qu'on ajoute une machine au réseau. Voici ce à quoi il ressemble dans le cas du réseau mis en place : les lignes sont au format `<IP> <nom> [<alias>]`
/etc/hosts

```
# /etc/hosts
# Cette ligne est nécessaire et se rapporte à l'interface lo dans
# /etc/network/interfaces.
127.0.0.1       localhost

# Les ordinateurs du réseau
192.168.1.1        julien
192.168.1.2        tux
```

Les serveurs DNS sont utilisés pour résoudre les noms sur Internet. Le fichier /etc/resolv.conf(5) de ***julien*** a été modifié lors de la connexion à Internet et contient les informations nécessaires : il suffit de le recopier sur ***tux*** en ne gardant que les lignes commençant par "nameserver". Dans le cas de Télédisnet, il est comme ceci :
/etc/resolv.conf

```
search teledisnet.be
nameserver 217.117.32.3
nameserver 194.7.1.4
```

.
Il reste à vérifier que /etc/host.conf(5) indique bien, lors d'une résolution de nom, de d'abord chercher dans les fichiers locaux (/etc/hosts) avant de faire une requête auprès des serveurs DNS.

Deuxième partie
Installation de services

5 Un serveur DNS

Configuration requise:
julien a besoin du package bind (le serveur DNS)

Le serveur DNS permettra de remplacer avantageusement le fichier /etc/hosts(5) si le réseau a une certaine taille, car il permet de centraliser les noms des machines.

5.1 Configuration du serveur

La configuration principale se fait dans le fichier /etc/bind/named.conf(5). Il est composé de sections ayant le format `ce qu'on configure { option1 valeur1; option2 valeur2; ...;};`
Les options peuvent être laissées telles quelles, c'est la configuration des zones qui est intéressante. Pour chaque zone, on définit comment le serveur doit la traiter (option type : master indique que le serveur possède les données pour résoudre la requête, hint indique que le serveur doit contacter d'autres serveurs pour résoudre la requête) et dans quel fichier se trouvent les données nécessaires à la résolution de la requête (option file).
Voilà le fichier sur ***julien*** (le début est semblable au fichier fourni par défaut, les ajouts sont à la fin) :
/etc/bind/named.conf

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind/README.Debian for information on the
// structure of BIND configuration files in Debian for BIND versions 8.2.1
// and later, *BEFORE* you customize this configuration file.
//

options {
        directory "/var/cache/bind";
};

// reduce log verbosity on issues outside our control
logging {
        category lame-servers { null; };
        category cname { null; };
};
```

```
// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};

// add entries for other zones below here

// Cette zone permet de connaître une adresse IP à partir d'un nom.
// On déclare le serveur compétent pour répondre aux requêtes pour
// le réseau local.
// Ce réseau est décrit dans le fichier /etc/bind/db.localnet
zone "localnet" {
        type master;
        file "/etc/bind/db.localnet";
};

// Cette zone permet d'obtenir un nom à partir d'une adresse IP.
// Pour obtenir son nom, on inverse la position des nombres de l'adresse
// du réseau pour lesquels le nombre correspondant du masque est à 255,
// et on ajoute .in-addr.arpa
```

```
// C'est le fichier /etc/bind/db.192.168.1 qui indique les correspondances
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192.168.1";
};
```

5.1.1 La zone "."

Cette zone permet de résoudre les noms sur Internet. Elle est de type hint : elle fera appel à d'autres serveurs pour résoudre les noms. La configuration se trouve, comme précisé dans la zone, dans le fichier /etc/bind/db.root. Pour le créer, il faut taper la commande `dig @a.root-servers.net > /etc/bind/db.root` (il faut être connecté à Internet pour taper cette commande). Le résultat est le suivant :
/etc/bind/db.root

```
; «» DiG 9.2.1 «» @a.root-servers.net
;; global options:  printcmd
;; Got answer:
;; -»HEADER«- opcode: QUERY, status: NOERROR, id: 54128
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13

;; QUESTION SECTION:
;.                                  IN        NS

;; ANSWER SECTION:
.                         518400        IN        NS        J.ROOT-SERVERS.NET.
.                         518400        IN        NS        K.ROOT-SERVERS.NET.
.                         518400        IN        NS        L.ROOT-SERVERS.NET.
.                         518400        IN        NS        M.ROOT-SERVERS.NET.
.                         518400        IN        NS        I.ROOT-SERVERS.NET.
.                         518400        IN        NS        E.ROOT-SERVERS.NET.
.                         518400        IN        NS        D.ROOT-SERVERS.NET.
.                         518400        IN        NS        A.ROOT-SERVERS.NET.
.                         518400        IN        NS        H.ROOT-SERVERS.NET.
.                         518400        IN        NS        C.ROOT-SERVERS.NET.
.                         518400        IN        NS        G.ROOT-SERVERS.NET.
.                         518400        IN        NS        F.ROOT-SERVERS.NET.
.                         518400        IN        NS        B.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
J.ROOT-SERVERS.NET.        3600000        IN        A        192.58.128.30
K.ROOT-SERVERS.NET.        3600000        IN        A        193.0.14.129
```

```
L.ROOT-SERVERS.NET.        3600000        IN        A        198.32.64.12
M.ROOT-SERVERS.NET.        3600000        IN        A        202.12.27.33
I.ROOT-SERVERS.NET.        3600000        IN        A        192.36.148.17
E.ROOT-SERVERS.NET.        3600000        IN        A        192.203.230.10
D.ROOT-SERVERS.NET.        3600000        IN        A        128.8.10.90
A.ROOT-SERVERS.NET.        3600000        IN        A        198.41.0.4
H.ROOT-SERVERS.NET.        3600000        IN        A        128.63.2.53
C.ROOT-SERVERS.NET.        3600000        IN        A        192.33.4.12
G.ROOT-SERVERS.NET.        3600000        IN        A        192.112.36.4
F.ROOT-SERVERS.NET.        3600000        IN        A        192.5.5.241
B.ROOT-SERVERS.NET.        3600000        IN        A        128.9.0.107

;; Query time: 114 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net)
;; WHEN: Wed Jan  8 14:13:28 2003
;; MSG SIZE  rcvd: 436
```

5.1.2 La zone “localhost”

Cette zone permettra à chaque ordinateur de résoudre le nom ***localhost*** en `127.0.0.1`. La configuration se trouve dans /etc/bind/db.local. Ce fichier est composé de lignes (elles peuvent être éclatées sur plusieurs lignes). Les lignes commençant par “ ;” sont des commentaires. Chaque fichier de zone commence par une ligne `@ IN SOA ....` @ désigne la zone qui est décrite par le fichier (elle est donnée dans /etc/bind/named.conf(5) : ici, c’est localhost), IN désigne le type de la zone (zone internet, sauf très rares exceptions, toutes les zones sont de type IN), SOA signifie Start Of Authority : la ligne donne les informations sur la manière de gérer la zone. Ensuite, il y a le serveur DNS qui contient le fichier de référence de la zone (
!!! Les points après les noms de machines ont une importance : sans eux, un nom de domaine serait suffixé
). Ici, c’est localhost : on est en train d’écrire ce fichier. Ensuite, l’adresse e-mail de celui qui est responsable de la zone, où l’arobase est remplacé par un point (ici : root@localhost). Puis, entre parenthèses, on trouve divers paramètres sans importance pour une configuration simple.
Ensuite, la ligne `@ IN NS localhost.` indique le serveur DNS (NS) à consulter pour résoudre un nom de la zone. Ici encore, c’est l’ordinateur local. Ensuite, on trouve une ligne `@ IN A 127.0.0.1` qui permettra de convertir ***localhost*** en `127.0.0.1` (pour rappel, @ désigne la zone décrite par le fichier, en l’occurence localhost).
/etc/bind/db.local

```
;
; BIND data file for local loopback interface
;
$TTL        604800
```

```
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS       localhost.
@       IN      A       127.0.0.1
```

5.1.3 La zone “127.in-addr.arpa”

Les zones de ce type (dont le nom se termine en .in-addr.arpa) servent à faire la conversion inverse (convertir une adresse IP en un nom). Avant .in-addr.arpa, il faut mettre les premiers nombres des adresses IP concernées (ceux qui correspondent à 255 dans le masque de sous-réseau) en sens inverse.
Dans le fichier de zone correspondant, on placera des lignes avec les adresses de la zone, en mettant les quatre nombres en sens inverse et en omettant ceux qui figurent dans le nom de la zone. Ensuite vient IN qui spécifie qu’on travaille sur une zone Internet, PTR, qui indique qu’on va donner un nom à une adresse, puis le nom correspondant à l’adresse (terminé par un point). Cela sera plus clair dans d’autres fichiers.
/etc/bind/db.127

```
;
; BIND reverse data file for local loopback interface
;
$TTL        604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS       localhost.
1.0.0       IN       PTR        localhost.
```

5.1.4 Les zones “0.in-addr.arpa” et “255.in-addr.arpa”

Ces zones servent pour le broadcast (quand on envoie des données à toutes les machines d’un réseau). On peut faire sans, mais comme elles sont placées dans la configuration par défaut, autant en profiter. Voici les fichiers de zone.

/etc/bind/db.0

```
;
; BIND reverse data file for broadcast zone
;
$TTL        604800
@       IN      SOA        localhost. root.localhost. (
                           1               ; Serial
                      604800               ; Refresh
                       86400               ; Retry
                     2419200               ; Expire
                      604800 )        ; Negative Cache TTL
;
@       IN      NS         localhost.
```

/etc/bind/db.255

```
;
; BIND reverse data file for broadcast zone
;
$TTL        604800
@       IN      SOA        localhost. root.localhost. (
                           1               ; Serial
                      604800               ; Refresh
                       86400               ; Retry
                     2419200               ; Expire
                      604800 )        ; Negative Cache TTL
;
@       IN      NS         localhost.
```

5.1.5 Les zones “localnet” et “1.168.192.in-addr.arpa”

C’est ici que ça devient propre au réseau local. Elles se comportent comme les zones “localhost” et “127.in-addr.arpa” et ce qui s’y trouve est valable ici. Voici les fichiers de zones qui permettront de repérer les ordinateurs du réseau local, inclus par la même occasion dans un domaine nommé “localnet” (
!!! le top level domain (ce qu’il y a après le dernier point) ne doit pas exister sur Internet
).
/etc/bind/db.localnet

```
;
; Fichier de données BIND pour le réseau local
```

```
; Les commentaires commencent par ";"
$TTL        604800
@        IN        SOA        julien.localnet. root.julien.localnet. (
                               1               ; Serial
                          604800               ; Refresh
                           86400               ; Retry
                         2419200               ; Expire
                          604800 )        ; Negative Cache TTL
;
@        IN        NS        julien.localnet.
; les ordinateurs de la zone
julien        IN        A        192.168.1.1
tux        IN        A        192.168.1.2
```

/etc/bind/db.192.168.1

```
;
; fichier de données BIND inverse
;
$TTL        604800
@        IN        SOA        julien.localnet. root.julien.localnet. (
                               1               ; Serial
                          604800               ; Refresh
                           86400               ; Retry
                         2419200               ; Expire
                          604800 )        ; Negative Cache TTL
@        IN        NS        julien.localnet.
; les ordinateurs de la zone
1        IN        PTR        julien.localnet.
2        IN        PTR        tux.localnet.
```

5.2 Configuration des clients

Les clients auront à modifier leur fichier /etc/resolv.conf(5) pour utiliser le serveur DNS hébergé par ***julien*** avant ceux de Télédis, et pour utiliser le nom de domaine localnet[5] :
/etc/resolv.conf

```
search localnet
nameserver 192.168.1.1
nameserver 217.117.32.3
nameserver 194.7.1.4
```

[5]La ligne “search localnet” indique par exemple que lorsqu’on cherche l’adresse ip correpondant à “julien”, il faut en fait rechercher “julien.localnet”.

tux peut donc maintenant se passer du fichier /etc/hosts(5)

5.3 Configuration du serveur

La même opération est à réaliser sur ***julien***, mais le problème est que lors de la connexion à Internet, Télédis met à jour automatiquement le fichier /etc/resolv.conf(5) avec les paramètres nécessaires pour l'accès à Internet. Heureusement, dhclient(8) permet d'écraser ces paramètres ou d'en rajouter. Il suffit de créer le fichier suivant pour qu'à chaque connexion les paramètres soient corrects.
/etc/dhclient.conf

```
# /etc/dhclient.conf file for /sbin/dhclient included in Debian's
#        dhcp-client package.
#

# eth0 est l'interface servant à la connexion à Internet
interface "eth0" {
        supersede domain-name "localnet";
        prepend domain-name-servers 127.0.0.1;
}
```

Cela va donc changer le domaine en "localnet" et ajouter le serveur DNS local à la liste des serveurs DNS. En attendant la prochaine reconnexion, on peut éditer manuellement /etc/resolv.conf(5)

6 Un serveur DHCP

> Configuration requise:
> le package dhcp doit être installé sur ***julien***. ***tux***, lui, aura besoin d'un client dhcp (package dhcp-client, pump ou dhcpcd)

Toujours dans un esprit de centralisation de la configuration, le serveur DHCP est un pas supplémentaire. Télédisnet, plutôt que de demander à ses clients d'encoder manuellement les paramètres réseaux, les attribue automatiquement grâce à son serveur DHCP. Nous pouvons faire de même sur le réseau local, et cela aura aussi l'avantage de configurer automatiquement les machines Windows.
Le package dhcp fournit un serveur dhcp qui s'appelle dhcpd. Il se configure à l'aide d'un seul fichier, dhcpd.conf(5). La configuration exemple définit une plage d'adresses de `192.168.1.10` à `192.168.1.20` allouées à n'importe quel ordinateur du réseau, et donne toujours à tux la même adresse `192.168.1.2` ; tux est reconnu grâce à l'adresse MAC de sa carte réseau, qu'on obtient grâce à la commande

```
ifconfig eth0|grep eth0|awk '{print $5;}'
```

(à taper sur tux) [6].

[6] Une astuce plus amusante consiste à consulter le cache ARP d'une machine du réseau ayant communiqué avec la machine dont on désire connaître l'adresse MAC. Ce cache peut être lu dans /proc/net/arp.

/etc/dhcpd.conf

```
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
option domain-name "localnet";
option domain-name-servers 192.168.1.1;

option subnet-mask 255.255.255.0;

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.20;
  option broadcast-address 192.168.1.255;
  option routers 192.168.1.1;
  host tux {
    hardware ethernet FE:FD:00:00:00:01;
    fixed-address 192.168.1.2;
    option host-name "tux";
  }
}

# Il faut une déclaration subnet pour chaque réseau sur lequel se trouve
# l'ordinateur.  Ces coordonnées correspondent aux paramètres donnés par
# Télédis
# Des alternatives à cette solution (si l'adresse IP change) sont de démarrer
# le serveur dhcp avant la connexion à Internet ou de placer le serveur dhcp
# sur un autre ordinateur du réseau.
subnet 217.117.58.0 netmask 255.255.254.0 {
}
```

7 Un serveur FTP

Configuration requise:
Il faut installer un serveur FTP sur ***julien***. Le package proftpd fera l'affaire. D'autres serveurs disponibles sont ftpd, wu-ftpd (réputé peu sécurisé), vsftpd (très sécurisé), lukemftpd, mudd-leftpd . . .

Une fois proftpd installé, il fonctionne. Les utilisateurs ayant un compte sur julien ont accès à leur répertoire personnel et tout le monde a un accès en lecture seule dans /home/ftp avec le login ftp ou anonymous et une adresse e-mail comme mot de passe. Le fichier de configuration

est /etc/proftpd.conf(5), sa syntaxe ressemble à celle du fichier de configuration d'Apache et il est très bien commenté.

8 Telnet

Configuration requise:
julien a besoin du package telnetd, qui fournit le serveur Telnet et permet d'accepter des connexions à distance ; ***tux***, lui, a besoin du package telnet, qui contient le client utilisé pour se connecter.

Le client et le serveur sont directement utilisables après leur installation. Pour se connecter à un ordinateur, taper

```
telnet nom_de_la_machine
```

, puis entrer son login et son mot de passe sur cette machine. Attention, telnet ne convient que dans le cadre de réseaux locaux, car la communication n'est pas cryptée et il est facile de l'espionner et de récupérer les mots de passes. C'est pour cela qu'il est recommandé d'utiliser SSH à la place.
NB: La plupart des protocoles ne sont pas cryptés : HTTP, FTP, POP, SMTP ...

9 SSH

Configuration requise:
Le package ssh contient à la fois le serveur et le client ssh, il faut donc l'installer sur ***julien*** et ***tux***.

L'usage basique est le même, sauf qu'on utilise la commande ssh(1) à la place de telnet(1). Il est possible aussi de s'authentifier non pas par le login et le mot de passe, mais par une paire de clés. En plus de la connexion à distance, SSH permet aussi la copie de fichiers grâce à la commande scp(1), dont la syntaxe est

```
scp source destination
```

, où les paramètres source et destination sont du type nom_du_fichier pour un fichier local et utilisateur@machine :/chemin/vers/fichier pour un fichier sur la machine distante.

10 Applications X distantes et terminaux X

Configuration requise:
Il suffit d'un serveur X en état de fonctionnement sur l'ordinateur où on désire afficher un programme, ***tux*** en l'occurence.

10.1 Exécuter une application graphique à distance

Les applications graphiques sous Linux réalisent leurs affichages en se connectant à un serveur X, qui contrôle un moniteur, un clavier et une souris. Il envoie au programme les mouvements de la souris et les frappes au clavier, et affiche à l'écran ce que le programme désire afficher. L'intérêt est que le serveur X ne doit pas obligatoirement tourner sur l'ordinateur qui exécute

l'application. Cela permet donc d'exécuter un programme sur une machine puissante et de travailler dessus sur une machine moins puissante (la puissance nécessaire est de l'ordre du 486DX 66 MHz + 16 MB de RAM) ou de lancer une application graphique se trouvant sur un ordinateur sans y avoir d'accès physique. Le principe est donc tout simple : dire à un programme d'utiliser non pas le serveur X qui est installé sur la machine locale, mais un serveur X se trouvant sur une autre machine du réseau, mais auparavant, il faut dire au serveur X d'accepter les connexions depuis les ordinateurs extérieurs en tapant la commande

```
xhost +nom_de_la_machine
```

dans une console lancée depuis le serveur X qui fera j'affichage. Ensuite, sur une console quelconque de l'ordinateur qui exécutera le programme (une console virtuelle, un xterm ou une console telnet/ssh), on positionne la variable d'environnement $DISPLAY. Si le serveur X se trouve sur la machine qui exécute le programme, elle contient simplement la valeur " :0" (ou " :1" s'il y a un second serveur X sur la machine). Si le serveur X se trouve sur une machine distante, on préfixe la valeur du nom de la machine ou de son adresse IP (exemple : "tux :0"). Pour rappel, sous bash, on spécifie une variable d'environnement par la commande

```
export VARIABLE=valeur
```

, soit

```
export DISPLAY=tux :0
```

. Maintenant, on peut lancer l'application dans cette console.
NB: Pour retirer le droit d'utiliser le serveur X, taper

```
xhost -nom_de_la_machine
```

NB: la commande xhost(1) offre peu de sécurité puisqu'elle autorise l'accès au serveur X depuis une machine sans tenir compte des utilisateurs. Quelqu'un de connecté à la machine autorisée à utiliser le serveur X pourrait donc envahir l'écran.

10.2 Lancer une session graphique à distance

> Configuration requise:
> xdm ou une version améliorée (kdm ou gdm), kdm servira dans l'exemple

Ici, on ne se contente plus de lancer une application à distance, mais on ouvre carrément une session sur la machine distante, avec gestionnaire de fenêtre … Cela est rendu possible par le protocole XDMCP. Sur ***julien***, il faut éditer le fichier /etc/kde2/kdm/kdmrc et y mettre la ligne `Enable=true` dans la section [Xdmcp]. Il faut également éditer le fichier /etc/kde2/kdm/Xaccess et y lister les ordinateurs pouvant utiliser le serveur (ou une astérisque pour spécifier tous les ordinateurs). Après avoir redémarré kdm, cela a pour effet d'autoriser les connexions à distance. Il suffit alors sur ***tux*** de taper

```
X -query julien
```

pour que l'écran de KDM de ***julien*** s'affiche sur ***tux***.

11 NFS

Configuration requise:
Le serveur (qui partage les fichiers) doit posséder un kernel compilé avec le support du serveur NFS (cette option est généralement activée sur toutes les distributions) et le package nfs-kernel-server.
NB: Il existe également un autre serveur NFS se trouvant entièrement dans l'espace utilisateur (ne requiert pas de code particulier dans le kernel)
Le serveur NFS permet de rendre accessible des répertoires à d'autres ordinateurs. Pour accéder à un répertoire ainsi exporté, il suffit de le monter comme un système de fichiers classique.

Quel que soit le serveur choisi, la configuration se trouve dans le fichier /etc/exports(5). Il peut ressembler à ceci :
/etc/exports

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).

# Exporter le répertoire /pub vers tous les ordinateurs, en lecture seule (ro)
# et en donnant à tous les utilisateurs les droits de l'utilisateur
# anonymous (all_squash)
/pub *(ro,all_squash)

# Exporte le répertoire /home/utilisateur vers tux.  Les droits d'accès sont
# ceux donnés par le numéro d'utilisateur et de groupe sur la machine depuis
# laquelle il se connecte.
/home/utilisateur tux(rw)
```

Il faut faire attention lorsqu'on exporte un répertoire : les droits d'accès aux fichiers exportés sont déterminés par le numéro d'utilisateur et le numéro de groupe sur la machine qui monte le répertoire, machine sur laquelle on peut n'avoir aucun contrôle (exemple : un portable qui se connecte sur le réseau).
Si on ne fait pas attention, on risque de rendre accessibles des données qu'on ne souhaite pas rendre lisibles.
Pour monter un répertoire NFS sur tux, il faut y taper une ligne du genre `mount -t nfs julien :/pub /mnt`. Bien sûr, on peut rajouter une ligne dans /etc/fstab pour autoriser les utilisateurs à monter eux-mêmes le répertoire, ou bien utiliser un automonteur qui démontera automatiquement le répertoire après usage.

12 Suggestion : un terminal graphique sans disque dur

Il est possible de transformer un ordinateur en terminal X diskless (sans disque dur) : il démarre sur une disquette et affiche l'écran de connexion de KDM d'une machine du réseau. Cela

constitue un bel exercice puisqu'il implique l'utilisation de plusieurs services. Voici quelques indications pour y parvenir, mais des lectures supplémentaires seront nécessaires : pages de man, documentations de linux (/usr/src/linux/Documentation) ...

– Le terminal est un système Linux normalement constitué : il a un kernel et un système de fichiers racine (/)
– Le système de fichiers racine se trouve sur un ordinateur du réseau et est monté par NFS. Cela implique la sélection de quelques options lors de la compilation du kernel :
  – Support pour le système de fichiers racine par NFS
  – Autoconfiguration par DHCP au démarrage
– Ce système est configuré pour lancer `X -query julien` au démarrage
– Il faut un serveur DHCP pour donner au kernel ses paramètres réseaux. D'autres paramètres comme l'endroit du système de fichiers racine sont passés par DHCP et doivent être mentionnés par le fichier de configuration.
– Le kernel est placé sur une disquette. Il faut lui dire d'utiliser un système de fichiers par NFS soit par la commande rdev, soit en installant lilo sur la disquette. Le pilote de la carte réseau doit être compilé dedans, pas en module.

La séquence de démarrage du terminal X serait donc

1. chargement du kernel depuis la disquette, avec ou sans lilo
2. le kernel s'auto-configure par DHCP, reçoit son adresse IP, l'adresse de son système de fichiers racine et les autres paramètres nécessaires.
3. le kernel configure la carte réseau et monte sa racine par NFS
4. le processus de boot normal se déroule (il comprend la connexion à un kdm/xdm/gdm du réseau) et l'écran de connexion s'affiche.

13 Partager des fichiers avec Windows (SAMBA)

> Configuration requise:
> Pour partager des fichiers et des imprimantes par SAMBA, il faut installer le package samba (ce rôle sera dévolu à ***julien***. Pour accéder aux ressources partagées, il faut un client : smbclient. Ce package sera installé à la fois sur ***julien*** et sur ***tux***. Le package smbfs permettra d'accéder à des répertoires exportés en les montant sur l'arborescence locale.

SAMBA se configure grâce au fichier /etc/samba.conf(5). Les commentaires peuvent être précédés soit d'une dièse, soit d'un point-virgule. Les paramètres globaux sont mentionnés après une ligne [global] ; les plus importants sont

1. le workgroup (il doit être le même pour tous les ordinateurs du réseau).
2. security, qui peut prendre la valeur user ou share selon que les droits d'accès dépendent de l'utilisateur qui veut utiliser la ressource ou la ressource. Dans l'exemple, share a été utilisé.
3. encrypt password : il doit valoir true si le réseau comporte des machines Windows 98, ME, 2000 ou XP ; il est possible de ne pas encrypter les mots de passes, mais cela nécessite de changer la base de registre sur les machines Windows. Par défaut, Windows 95 et NT 4 ne cryptent pas les mots de passes.

Ensuite, chaque ressource exportée est décrite de la manière suivante : `[nom_ressource] comment = Description de la ressource path = /chemin/vers/ressource read only = booléen (yes ou no) public = yes` Ce paramétrage est basique, il y a évidemment moyen de faire mieux. Finalement, le fichier /etc/samba/smb.conf ressemble à ceci :
/etc/samba/smb.conf

```
[global]

# Change this for the workgroup/NT-domain name your Samba server will part of
   workgroup = DEBONA

# server string is the equivalent of the NT Description field
   server string = %h server (Samba %v)

   invalid users = root

# This tells Samba to use a separate log file for each machine
# that connects
   log file = /var/log/samba/log.%m

# Put a capping on the size of the log files (in Kb).
   max log size = 1000

# We want Samba to log a minimum amount of information to syslog. Everything
# should go to /var/log/samba/log.{smb,nmb} instead. If you want to log
# through syslog you should set the following parameter to something higher.
   syslog = 0

# "security = user" is always a good idea. This will require a Unix account
# in this server for every user accessing the server. See
# security_level.txt for details.
;   security = user
        security = share

# You may wish to use password encryption. Please read ENCRYPTION.txt,
# Win95.txt and WinNT.txt in the Samba documentation. Do not enable this
# option unless you have read those documents
   encrypt passwords = true


# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
# You may want to add the following on a Linux system:
```

```
#         SO_RCVBUF=8192 SO_SNDBUF=8192
   socket options = TCP_NODELAY



# This will prevent nmbd to search for NetBIOS names through DNS.
   dns proxy = no

   obey pam restrictions = yes

#======================= Share Definitions =======================

[homes]
   comment = Home Directories
   browseable = no

# By default, the home directories are exported read-only. Change next
# parameter to 'yes' if you want to be able to write to them.
   writable = no

# File creation mask is set to 0700 for security reasons. If you want to
# create files with group=rw permissions, set next parameter to 0775.
   create mask = 0700

# Directory creation mask is set to 0700 for security reasons. If you want to
# create dirs. with group=rw permissions, set next parameter to 0775.
   directory mask = 0700

[divers]
        comment = Données diverses
        path=/home/data
        read only = yes
        public = yes
[incoming]
        comment = Répertoire de dépôt
        path=/home/incoming
        read only = no
        public = yes
```

Par ailleurs, l'utilisation des mots de passes cryptés nécessite la création d'un fichier de mots de passes (/etc/samba/smbpasswd) pour SAMBA. L'installation du package samba propose de créer ce fichier. Si cela n'a pas été fait, il est encore temps de le faire en tapant en root
`dpkg-reconfigure samba`

, qui fera apparaître un dialogue permettant de le créer à partir des mots de passes du système. Pour les utilisateurs qui seraient ajoutés par la suite grâce à la commande smbpasswd(8) suivie du nom de l'utilisateur à créer/changer. La commande demande ensuite d'entrer le mot de passe à attribuer à l'utilisateur.

A A propos de ce document

Ce document a été écrit en LaTeX sous VIM. Pour en obtenir le source, allez sur http ://debona.dyndns.org.

B Licence

Table des matières