Doc Concentrée pour Linux

par Antoine Levavasseur, levavass@efrei.fr V 0.63, le 3 Juin 1998

Cette documentation explique de façon concise et claire une manière (parmi d'autres) de configurer sa machine Linux (basé sur la RedHat) ainsi que quelques services réseau tels que NYS, NFS, WWW, Samba, NFS-Root, PPP...

Contents

1 Introduction

Ce document regroupe les réponses aux questions que se posent souvent les débutants sous Linux lorsqu'ils tentent de configurer leur machine et les différents services qui s'y rapportent.

Il pourra également être utile aux utilisateurs plus expérimentés, puisqu'il servira d'aide-mémoire et de guide-rapide lors de la configuration de fonctionnalités nouvelles ou déjà connues.

1.1 Ce que ce document est, et ce qu'il n'est pas

J'ai constaté que les HOWTO sont souvent très détaillés (et c'est normal) mais ne vont pas assez à l'essentiel. C'est pourquoi j'ai rédigé ce document en essayant de pas trop rentrer dans les détails, mais de fournir à chaque fois des informations suffisantes pour la majorité des utilisations et des solutions pour les problèmes les plus fréquents.

Cependant, il n'est pas question de remplacer les *HOWTO* <http://www.freenix.fr/linux/HOWTO/> ou le *guide du ROOTard Linux* <http://www.linux-france.com/article/grl/index.html> qui restent des documents de référence dont la lecture permet de résoudre les problèmes plus ardus et de découvrir de façon plus complète le domaine traité.

Ce document est entièrement basé sur les distribution RedHat 4.0 et suivantes, c'est un parti pris destiné à faciliter la lecture. Il ne tient donc aucun compte des éventuelles variations de structure de répertoires, noms de fichiers, et format des fichiers de configuration que présentent les autres distributions.

Les objectifs de cette doc sont de permettre de configurer votre machine Linux et quelques services sans trop se casser la tête, en profitant d'astuces et de conseils connus. Que vous soyez débutant ou utilisateur expérimenté avec Linux, j'espère qu'ici vous apprendrez de nouvelles choses, qui vous permettront de gagner, un jour où l'autre, un peu de temps.

1.2 Remarques diverses

Je ne suis évidemment pas infaillible, et des erreurs se sont probablement glissées dans ce document, il doit aussi y avoir certaines remarques qui méritent des corrections ou des ajouts. N'hésitez donc pas à m'envoyer vos remarques à l'adresse spécifiée plus bas.

Ce document est encore bien incomplet, d'ici peu j'espère pouvoir y ajouter un chapitre concernant le DNS ainsi que DHCP, Sendmail...

Ce document est désormais écrit en SGML (exactement comme les HOWTO). Cela permet des conversions vers quasiment tous les formats dont HTML, PostScript, texte ASCII, LaTeX et j'en passe...

1.3 Aspect Légal - Copyright

Ce document est public ou freeware. L'objectif est donc qu'il soit diffusé le plus largement possible. Je vous encourage donc vivement à le faire circuler, à la condition qu'il soit conservé dans son intégralité et non modifié, et qu'aucun bénéfice financier n'en soit tiré.

1.4 Où trouver ce document et qui le maintient ?

Ce document est maintenu par Antoine Levavasseur levavass@efrei.fr n'hésitez pas à m'envoyer vos suggestions, remarques, articles. Toutes les remarques sont bonnes pour parvenir à faire évoluer ce document. Les nouvelles versions de ce document se trouvent sur la page Web :
http://web.efrei.fr/~levavass/doc_concentree/

1.5 Autres sources d'informations

Les sources d'informations concernant Linux sont très nombreuses. Je saurais toutefois conseiller quelques références disponibles en-ligne :

- Guide du ROOTard pour Linux, document inévitable de la documentation Linux. Très complet. ftp://ftp.lip6.fr:/pub/linux/french/docs/ ou bien http://www.freenix/linux/Guide/
- Les HOWTOs, autres documents de référence pour configurer les différents services Linux. ftp://ftp.lip6.fr:/pub/linux/french/docs/HOWTO ou bien http://www.freenix.fr/linux/HOWTO/
- Et un des sites les plus actifs pour les documentations linux en français. http://www.linux-france.com/

Mais également des livres :

- *Le système Linux*
  <http://www.editions-oreilly.fr/international/france/sysad/runux2.html> par Matt Welsh & Lar Kaufman (618 pages, environ 280FF T.T.C.) chez O'Reilly. Traduction de René Cougnenc et Manuel Makarévitch.
- *Administration Réseau sous Linux*
  <http://www.editions-oreilly.fr/international/france/sysad/linag.html> par Olaf Kirch (336 pages, environ 220FF T.T.C.) chez O'Reilly. Traduction de René Cougnenc.

1.6 Remerciements

Je tiens à remercier toutes les personnes qui ont participé (directement ou indirectement) à l'élaboration de ce document :

Laurent Bertin <bertin@efrei.fr>
Thomas Parmelan <parmelan@efrei.fr>
Ludovic Bailly <bailly@efrei.fr>
Jean Charles Delepine <delepine@lan.univ-lyon1.fr>
Nat Makarévitch <nat@linux-france.com>

2 Installation de Linux

L'installation de la Redhat est suffisamment simple pour ne pas mériter de plus amples explications. Je fournirai toutefois quelques précisions sur 3 questions souvent posées.

2.1 Partitionnement des disques

```
Linux Swap (82)
   Taille conseillée : 32-64 Mo
   Taille maximale possible : 120 Mo par partition (rarement utile)

Linux Native (83)
   Taille minimale sans X : 80 Mo
   Taille minimale avec X : 120 Mo
   Taille conseillée avec X : 300 Mo
```

Mais l'espace disque n'est jamais suffisant, alors une taille de 800 Mo n'est vraiment pas ridicule !

Il faut préciser que certains utilisateurs préfèrent séparer par des partitions distinctes les répertoires `/home, /usr, /usr/local` pour des raisons pratiques. En effet, cela permet de conserver à coup sûr les comptes et logiciels entre 2 réinstallations.

2.2 Quelques info sur LILO

Il est possible d'installer lilo à 2 endroits sur le disque dur. Soit dans le MBR, soit dans le premier secteur de la partition root.

Si un problème survient du fait d'un virus ou de mauvaise manipulation, une commande non documentée du fdisk de MS-DOS permet de restaurer le MBR dans sa forme originelle : la commande est `fdisk /mbr`. Ensuite pour réamorcer Linux il faudra passer par une disquette bootable Linux, qu'il ne faudra pas oublier de créer.

Dans le cas ou lilo est placé dans le premier secteur de la partition Linux, il suffira d'activer la partition Linux avec fdisk pour que tout rentre dans l'ordre.

2.3 Créer une disquette bootable pour récupérer une machine en peine

Pour cela, la méthode un violente mais efficace : prendre un noyau qui fonctionne correctement (n'oubliez pas le support SCSI si votre disque dur est de ce type).

Copiez-le dans un autre endroit, et faites un

```
rdev nom_du_noyau /dev/hda1
```

Cela permet de spécifier où se trouve la partition root du système, évidemment si votre partition root Linux est différente de `hda1` mettre la valeur correcte.

Entrer la commande `rdev -R nom_du_noyau 1` permettra de monter la partition root en read only et permettera à fsck de travailler...

Ensuite faire un petit

```
cp nom_du_noyau /dev/fd0
```

Et la disquette de boot est prête. Attention elle ne peut être utile que si le problème vient d'un lilo endommagé ou d'un noyau un peu grincheux, mais elle sera inefficace si la partition root ne contient pas la suite des fichiers nécessaires au boot.

3 Différents fichiers de configuration

Une fois l'installation de la Redhat terminée, je propose de modifier quelques fichiers pour améliorer la convivialité et adapter votre système à vos besoins :

3.1 /etc/bashrc

On remplace W par w pour que le prompt du shell affiche le chemin complet

```
PS1="[\u@\h \w]$"
```

De plus on rajoute 2 alias pour que la commande **ls** soit en couleur et que la commande **l** liste en couleur et avec toutes les infos

```
alias ls='ls -NF --color=tty'
alias l='ls -NFail --color=tty'
```

3.2 /etc/profile

On remplace W par w pour que le prompt du shell affiche le chemin complet

```
PS1="[\u@\h \w]$"
```

Il faut mieux le faire dans les 2 fichiers (*profile* et *bashrc*) pour s'assurer qu'il fonctionne correctement. Chaque utilisateur peut bien entendu avoir d'autres paramètres en modifiant son fichier *.bashrc* et *.bash_profile*.

Pour que le programme `less` affiche correctement les accents, ajouter également :

```
export LESSCHARSET=latin1
```

IMPORTANT

En général le nom du répertoire courant (`./`) n'est pas inclus dans le chemin de recherche des exécutables (le `PATH`). C'est parfois agaçant, car il parait parfois impossible d'exécuter un fichier alors qu'avec un `ls`, il est là sous nos yeux avec les bon droits. Il est donc utile de rajouter dans la ligne PATH du fichier .bash_profile du répertoire home de l'utilisateur un :./. Voici un exemple :

```
PATH="$PATH:."
```

3.3 /etc/fstab

Le fichier `/etc/fstab` regroupe les différents périphériques qui peuvent être montés et ceux qui le seront par défaut au boot.

```
  Le mien ressemble à :
/dev/sda3     /           ext2     defaults 1 1
/dev/sda5     /dos        vfat     defaults 0 0
/dev/fd0      /mnt/floppy ext2     noauto 0 0
/dev/sda1     /win95      vfat     defaults 0 0
/dev/sdb4     /zip        auto     rw,user,noauto 0 0
/dev/hda      /cdrom      iso9660  ro,user,noauto,unhide 0 0
/dev/hdb1     /dur        auto     rw,user,noauto 0 0

none          /proc       proc     defaults

zeppelin:/users  /users   nfs     rw,rsize=8192,wsize=8192,noauto,user,intr,exec
```

Les lignes contenant l'option *noauto* ne seront pas montées lors du boot du système. L'option *user* indique que n'importe quel utilisateur pourra monter et démonter le périphérique.

Pour le montage en **nfs**, l'option `rsize=8192,wsize=8192` est facultative, mais permet d'augmenter sensiblement les performances.

Si l'édition de fichiers texte ne vous enchante qu'à moitié, la RedHat fournit également l'utilitaire *fstool* sous X qui permet de rajouter et supprimer les différents périphériques montés à l'aide de quelques clics de souris et autres boîtes de dialogues.

3.4 /etc/lilo.conf

Ce fichier décrit la configuration de lilo. Pour que ses modifications soient prises en compte, il faut ensuite lancer la commande **lilo**.

```
  La pédagogie par l'exemple :
boot=/dev/sda3
map=/boot/map
install=/boot/boot.b
prompt timeout=500
other=/dev/sda1
         label=d
         table=/dev/sda
image=/boot/vmlinuz
         label=l
         root=/dev/sda3
         initrd=/boot/initrd
         read-only
image=/boot/vmlinuz.old
         label=lold
         root=/dev/sda3
         initrd=/boot/initrd
         read-only
```

Si vous omettez l'option *prompt* et *timeout* qui permettent d'afficher un message et d'attendre le temps précisé par timeout, il suffit d'appuyer sur Ctrl+Shift et le Lilo prompt s'affichera malgré tout (utile pour tester un nouveau noyau) sinon il bootera sur la partition par défaut c'est à dire la première définie

Si vous avez une carte Ethernet et que Linux ne la détecte pas au démarrage vous pouvez utiliser la fonction *append* de LILO, rajouter :

```
append="ether=10,0x300,eth0"
```

où 10 est l'IRQ sur la carte Ethernet, 0x300 est l'adresse de la carte Ethernet et eth0 est l'interface qui la représentera.

En effet, Linux ne recherche les cartes Ethernet qu'à certaines adresses qui ne sont pas forcément celles où se trouve la votre, de plus cela peut être également utile si vous voulez utiliser plusieurs cartes Ethernet,

puisque Linux ne recherche et détecte par défaut qu'une seule carte Ethernet. Lire à ce propos le *BootPrompt HOWTO* <http://www.freenix.fr/linux/HOWTO/BootPrompt-HOWTO.html>.

L'option *append* est également utile si vous avez des problèmes à faire reconnaître votre disque dur. Mettre par exemple : `append = "hd=64,32,202"` (les nombres représentent C/S/H : Cylinders, Sectors, Heads) et votre disque sera correctement reconnu.

3.5 /etc/rc.d/

Ce répertoire contient les scripts lancés au démarrage de la machine. rc.sysinit est lancé en tout premier.

Ensuite les scripts définis dans init.d sont lancés par les liens contenus dans les répertoires rcN.d où N correspond au runlevel.

rc.local est lancé en dernier.

Si vous voulez automatiser certaines actions au lancement, vous pouvez les lancer dans le script rc.sysinit ou rc.local selon vos besoins.

Si vous trouvez que vous lancez trop de démons sur votre système, vous pouvez toujours supprimer les liens leur correspondant dans les différents répertoires rcN.d Mais attention certains sont parfois indispensables alors n'enlevez que ce que vous connaissez ou faites une sauvegarde !

Un utilitaire fourni avec la RedHat permet d'ailleurs de le faire de façon un peu plus visuelle. Celui-ci s'appelle *tksysv*, et il permet en quelques clics de souris de supprimer les démons qui paraissent encombrants sans avoir à parcourir les nombreux répertoires de `/etc/rc.d/`

3.6 /etc/syslog.conf

Le démon **syslogd** permet de récupérer différents messages du noyau, qui concernent des messages de lancement de certains programmes, des messages de loggin, et également des messages d'erreur.

Il est parfois intéressent de pouvoir consulter ces messages pour détecter une éventuelle erreur ou le dysfonctionnement d'un programme ou d'un service.

Pour cela le fichier syslog.conf permet avec des règles de choisir où ces messages doivent parvenir.

Je propose donc de rajouter en première ligne du fichier `/etc/syslog.conf` :

```
*.*                                     /dev/tty8
```

Elle va donc rediriger tout ces messages dans la console tty8.

Je rappelle que pour changer de console, il suffit de faire Alt+Fx, Fx correspondant à la touche de fonction comportant le numéro de la console que vous voulez atteindre. Sous X, il faut faire Ctrl+Alt+Fn.

4 Gestion des utilisateurs et des quotas disque

4.1 Ajouter des utilisateurs

Une fois votre Linux installé, il est temps de créer de nouveaux utilisateurs. C'est une mauvaise habitude que d'utiliser sa machine Linux sans cesse avec le compte root. Il est préférable d'avoir au moins un compte utilisateur sur lequel on travaillera. Si en cours d'utilisation certaines commandes demandent à être super-utilisateur, il sera encore temps de faire un **su -m**. Cela permet de limiter les fausses manipulations qui, du fait des privilèges root, peuvent avoir des conséquences fâcheuses.

Pour ajouter un nouvel utilisateur tonio :

```
$adduser tonio
```

puis il faut ensuite définir son mot de passe :

```
$passwd tonio
```

Parfois entre 2 séances de changement de mot de passe, il sera nécessaire de supprimer le fichier `/etc/.pwd.lock` qui empêche la modification des mots de passe pour des raisons de sécurité.

Pour ajouter, supprimer, modifier les paramètres des utilisateurs, vous pouvez également utiliser l'utilitaire *usercfg* fourni avec la RedHat qui sera idéal pour les adeptes de la souris ou les réfractaires du tout-clavier.

4.2 Mise en place des quota disque

Pour éviter la saturation du système et limiter l'espace alloué à chaque utilisateur, Linux permet de mettre en place des limites d'utilisation des ressources disques. Il faut tout d'abord que les utilitaires de quota disques soient installés, si ça n'est pas le cas utiliser *rpm* ou *glint*.
Il faut ensuite activer le support des quota dans votre noyau et éventuellement le recompiler. Le support des quotas est jusque là uniquement disponible avec le système de fichiers **ext2**.

Pour activer les quotas, il faut tout d'abord activer `usrquota` et `grpquota` respectivement pour les quotas utilisateurs et les quotas pour les groupes d'utilisateurs dans le fichier `fstab` :

```
/dev/sda4       /home    ext2    defaults,rw,usrquota,grpquota  0 0
```

La commande `quotacheck -avug` permet de vérifier la cohérence des informations des quotas. De plus elle permet lors de la première utilisation de créer les fichiers `quota.user` et `quota.group` dans lesquels sont stockés les quotas et par la suite de les remettre à jour.

Les quotas sont activés lors du lancement de `quotaon` généralement appelé automatiquement dans `rc.d`.

Pour modifier les quotas affectés à un utilisateur ou un groupe d'utilisateurs, il suffit d'utiliser `edquota -u user` et `edquota -g groupe`. Cette commande va permettre d'éditer un fichier texte dans lequel on pourra modifier les quotas. Les nouvelles valeurs seront prise en compte dès l'écriture du fichier.

Il existe 2 limitations : une limitation en nombre de fichiers et une limitation en taille (par blocs de 1 Ko).

Et ces limitations définissent 2 limites distinctes : la limite "douce" et la limite "dure". Si l'utilisateur dépasse la limite douce, il aura un message d'avertissement à chaque écriture de fichier. Si c'est la limite dure qui est atteinte, aucun nouveau fichier ne pourra être créé. La limite douce se transforme en limite dure par défaut au bout de 7 jours.

Chaque utilisateur peut obtenir l'état des quotas qui lui sont attribués grâce à la commande `quota`. De même en passant les bons paramètres le super-utilisateur pourra obtenir les mêmes informations.

5 Utilisation de RPM pour installer les logiciels

RPM : Redhat Package Management est un outil puissant spécifique à la distribution RedHat de Linux qui permet de regrouper dans un même fichier **.rpm** la totalité de ceux correspondant au logiciel. Cependant ce format est de plus en plus répandu, et d'autres distributions peuvent maintenant exploiter les fichiers **.rpm** de la même façon.

Ce système permet de gérer les mise à jours, les ajouts, les suppressions de programme très facilement tout en incluant la notion de dépendance, c'est à dire que certains logiciels ont besoin d'autres pour fonctionner. rpm se chargera alors d'installer automatiquement toutes les composantes nécessaire au bon fonctionnement de vos programmes.

Une interface au programme rpm existe sous X et elle est relativement pratique. Elle s'appelle *glint*. Cependant, il est parfois plus efficace d'utiliser directement la ligne de commande. Nous allons à travers quelques exemples montrer les possibilités de rpm.

Installer un nouveau package sur le système se fait avec :

```
rpm -i nom_du_package.v.i386.rpm
```

Pour mettre à jour ("upgrade") un package :

```
rpm -U nom_du_package.v.i386.rpm
```

Pour désinstaller(effacer) un package, il suffit d'utiliser :

```
rpm -e nom_du_package
```

Pour afficher la totalité des fichiers qui ont été modifiés depuis l'installation de package utiliser :

```
rpm -Va
```

Pour savoir le nom du package auquel appartient un fichier il suffit d'utiliser la commande :

```
rpm -qf nom_du_fichier
```

Pour obtenir des informations sur un package que vous venez de récupérer, il suffit de faire un petit :

```
rpm -qpi nom_du_package.rpm
```

La commande permettant de lister la totalité des fichiers qu'un package va installer est :

```
rpm -qpl nom_du_package.rpm
```

6 Recompiler et patcher son noyau

Recompiler son noyau est une tâche courante sous Linux, cela permet d'adapter celui-ci à de nouvelles caractéristiques de sa machine, ou bien de rajouter des fonctionnalités ou des corrections grâce à des patchs, puisque le principe même de Linux, en constant développement, est de pouvoir profiter quasiment aussitôt des mises à jour faites par les programmeurs.

6.1 Configurer puis compiler son noyau

Tout d'abord pour définir sa configuration, c'est-à-dire déterminer les options que vous voulez utiliser qui seront les plus adaptées à la machine, se positionner dans `/usr/src/linux` et vous pouvez utiliser 3 commandes différentes :

```
$make config
     Mode texte. Pose une rafale de question assez peu conviviale
$make menuconfig
     Mode texte couleur. Menu déroulants très utilisable
$make xconfig
     Tcl/Tk sous X. Pour les mordus de la souris, très bien aussi
```

Une fois terminé, il faut effacer les traces des compilations précédentes en faisant :

```
$make dep; make clean
```

Enfin la compilation proprement dite débute à la commande :

```
$make zImage
```

À ce stade, selon la machine, il faut attendre un peu : (486DX2 66 = 40mn, Pentium 120 = 20mn, Cyrix P166+ = 8 mn, PPro 200 = 4 mn) temps à titre indicatif, mais c'est un bon benchmark :-)

Si certaines parties du noyau sont activées en tant que modules, il faut également les compiler :

```
$make modules
```

Puis les installer :

```
$make modules_install
```

Récupérer ensuite le noyau dans `/usr/src/linux/arch/i386/boot/zImage`

Il faut ensuite le copier dans `/boot` et renommer les éventuels `vmlinuz`, `zImage` et autres selon sa convenance.

Éditer éventuellement le fichier /etc/lilo.conf pour rajouter au minimum une entrée pour l'ancien noyau et une pour le nouveau en cas de problème avec celui nouvellement compilé (ça n'est pas si rare !)

Pour finir lancer la commande lilo

rebooter : c'est prêt !

6.2 Appliquer un patch à son noyau

Pour patcher son noyau, la commande patch s'occupe de tout. Ainsi pour ajouter le patch nouveau.pch, se positionner dans `/usr/src/linux` : `$ patch -p1 < nouveau.patch`

Pour retirer proprement un patch, il suffit de l'appliquer une nouvelle fois. Le programme propose alors d'enlever les parties correspondantes.

7 Recherche de fichiers

Retrouver un fichier perdu dans l'arborescence n'est pas toujours facile. De plus, cela oblige le disque dur à ramer pendant de nombreuses secondes, pour parfois ne rien trouver. Heureusement, la commande **locate** permet de faire la recherche directement dans un fichier contenant la liste des fichiers de façon quasi-instantanée.

Cependant cette liste doit être créée et mise à jour régulièrement. C'est la commande **updatedb** qui s'en charge.

8 Utiliser Netscape (Navigator 3.x et Communicator 4.x) sous Linux

Les anciens programmes Netscape 3.x et Communicator 4.2 étaient programmées avec des anciennes libc et certains bugs les empêchent de fonctionner avec les nouvelles librairies. Il était nécessaire de récupérer les anciens fichiers et de passer quelques paramètres pour que tout fonctionne correctement.

Une description complète du processus à suivre est toujours disponible sur la Linux-Netscape Help Page qui fait office de HOWTO sur la question.

Une verison corrigée des bibliothèques responsables du dysfonctionnement est disponible sur Netscape fix

Cependant, la nouvelle orientation de Netscape vers le logiciel libre à permis de résoudre ce problème. On trouve maintenant une version rpm de Netscape fournie dans le répertoire `/contrib` de RedHat qui permet de l'installer très facilement.

Certains utilisateurs peuvent encore rencontrer quelques soucis au niveau des couleurs de l'application si leur serveur X fonctionne en mode 24 bits. Le résultat est que les icônes apparaissent en Noir et Blanc, mais cela n'a pas d'incidence sur le fonctionnement.

Il faut enfin préciser que les programmes de Netscape sont très (trop) sensibles aux problèmes réseaux. Ainsi vérifiez que votre `resolv.conf` est correctement configuré, et que le programme `route` rend correctement la main (sinon une route est sans doute mal configurée).

9 Le support réseau sous Linux

9.1 Support des interfaces réseau

Si vous avez spécifié des paramètres corrects lors de l'installation de la machine, le support des interfaces réseau doit déjà fonctionner.

Si ce n'est pas le cas, il n'est pas trop tard pour le faire. Tout d'abord le noyau doit supporter les interfaces réseau.

Ensuite vérifier que les interfaces sont bien supportées avec la commande **ifconfig**, si vous avez les interfaces loopback et Ethernet c'est ok.

Si vous avez des problèmes à faire reconnaître votre carte Ethernet essayez d'utiliser la fonction append dans le fichier `/etc/lilo.conf`

Les interfaces activées automatiquement au boot sont définies dans les fichiers du répertoire `/etc/sysconfig/network-scripts/`

Si vous ne voulez pas configurer les interfaces de façon automatique, vous pouvez bien sûr utiliser la commande ifconfig en faisant par exemple (cas d'une machine d'adresse IP 192.168.1.202) :

```
$ifconfig eth0 192.168.1.202
```

9.2 Ajout des routes

Une fois vos interfaces configurées correctement, il faut maintenant ajouter les routes, c'est à dire les adresses IP des ordinateurs avec lesquels vous voulez vous connecter. La commande **netstat -r** affiche les routes déjà configurées, il faut au moins avoir la route vers loopback.

```
Pour les ajouter de nouvelles routes faire :
    $route add 127.0.0.0
    $route add 192.168.1.202
```

De la même manière, ajouter les routes vers les ordinateurs vers lesquels vous voulez avoir accès en spécifiant leurs adresses IP. Cependant, vous n'êtes pas obligé de configurer une route pour chaque ordinateur. En ajoutant une route vers le sous-réseau (par exemple 192.168.1.0) tous les ordinateurs connectés au brin local qui ont une adresse commençant par 192.168.1 seront accessibles.

Pour vérifier que le réseau marche bien, il faut maintenant tester grâce à un petit **ping** en spécifiant l'adresse d'un ordinateur de votre réseau.

9.3 Spécifier le nom de la machine, puis des hôtes de votre réseau

Si vous voulez modifier le nom de votre ordinateur, il faudra modifier :

- la ligne HOSTNAME du fichier `/etc/sysconfig/network`
- le contenu du fichier `/etc/HOSTNAME`
- la ligne correspondante de votre fichier `/etc/hosts`
- et tous les autres endroits où vous l'avez spécifié dans un fichier de configuration...

Spécifier les adresses IP de chaque machine n'est évidemment pas très humain, on peut donc associer le nom des machines aux adresses IP, cela se fait dans le fichier /etc/hosts

```
  le mien ressemble à :
127.0.0.1          localhost
192.168.1.200      zeppelin
192.168.1.201      pantera
192.168.1.202      fresne
192.168.1.203      trixie
192.168.1.209      flupke
```

Ensuite toutes les manipulations qui concernent l'adresse IP pourront se faire en spécifiant le nom de la machine.

9.4 Regrouper des machines par sous-réseau

À partir d'un certain nombre de machine, le fichier hosts peut prendre une taille plutôt conséquente, c'est pourquoi il est possible de regrouper les machines par leurs adresses IP dans un sous-réseau auquel on donnera alors un nom.

Les sous-réseaux sont définis dans le fichier `/etc/networks` comme suit :

```
zep-net     192.168.1.0
autre-net   198.163.8.0
```

Le sous-réseau zep-net regroupe en fait toutes les machines commençant par 192.168.1, c'est-à-dire les machines définies plus haut dans le fichier /etc/hosts.

Cela simplifie ainsi beaucoup l'ajout des routes ou les autres manipulations que l'on fera alors pour le sous-réseau entier plutôt que machine par machine.

9.5 Automatiser la configuration des routes

Automatiser la configuration des routes au boot du système se fait dans le fichier `/etc/sysconfig/static-routes`. Il est généralement inexistant, il faut donc le créer. Cependant, cela n'est vraiment indispensable que pour ceux qui ont des ordinateurs repartis sur de nombreux sous-réseaux différents.

```
  voila toutefois un petit exemple :
eth0 host flupke
eth0 host zebulon
eth0 net  zep-net
```

la close host est utilisée pour d'autres ordinateurs.
la close net est utilisée pour les sous-réseaux.

Une fois cela terminé, le réseau doit marcher correctement, il est maintenant temps de commencer à lancer des services réseau...

10 NFS : le système de fichier réseau

Le service NFS (Network File System) permet de monter des périphériques d'hôtes distants et de les utiliser comme si ils faisaient partie de sa propre machine. Ce système est pratique et efficace. Il a juste un inconvénient, c'est qu'il est relativement lent. Sa mise en place est des plus simple.

On configure dans le fichier `/etc/exports` les répertoires qui seront autorisés à être montés par les machines clientes.

```
  Un petit exemple pour avoir une idée :

/users       fresne(rw) pantera(rw) flupke(rw)
/            fresne(ro)
/usr/local
```

Évidement, il est souhaitable pour des raisons de sécurité de ne pas permettre à tout le monde de monter des fichiers en lecture-écriture(rw).

Vous pouvez maintenant lancer sur le serveur les démons NFS qui sont `rpc.mountd` et `rpc.nfsd`. Bien entendu, il faut que le portmapper soit lancé.

Vous devez enfin autoriser la connexion en ajoutant une ligne dans le fichier

```
  hosts.allow du style :

ALL: 192.168.1.          ou bien
ALL: fresne, pantera
```

Le serveur est maintenant prêt, il n'y a plus qu'à essayer de monter un répertoire sur votre client.

```
   Essayer la commande :
mount -t nfs nom_du_serv:/rep_exporte /rep_local
```

Il faut tout de même que le noyau supporte le système de fichier nfs, vérifier en faisant un `cat /proc/filesystems`, une ligne `nodev nfs` doit apparaître, si ça n'est pas le cas, vous devez recompiler votre noyau.

11 NYS/NIS/YP ou les comptes distribués

Le service NYS permet de distribuer sur un certain nombre de machines des cartes. Ces cartes sont en fait les fichiers de configuration correspondant aux comptes utilisateurs, aux adresses des différentes machines...

Pratiquement, NYS permet en fait d'utiliser son compte utilisateur avec le même mot de passe et les mêmes paramètres sur n'importe quelle machine appartenant au réseau NYS. Pour cela, on exportera les comptes utilisateur par NFS, et on utilisera NYS pour partager les données et les paramètres sur tous les clients du réseau.

Une fois installé, tous les fichiers de configuration sont centralisés sur le serveur. Cela simplifie grandement les mises à jour, la maintenance, et les ordinateurs clients ont ainsi en local des fichiers minimum.

Les cartes sont créées à partir des fichiers du serveur :

```
/etc/passwd    -> carte des mots de passe
/etc/group     -> groupe des utilisateurs
/etc/hosts     -> les machines du réseau
/etc/networks  -> liste des sous-réseaux
```

11.1 Mettre en place le serveur

Avant tout, il faut spécifier à quel domaine NIS appartient la machine. Cela se fait par la commande domainname.

```
$domainname mon_domaine
```

Ce domaine est totalement différent du domaine DNS. Vous pouvez donc donner un nom de domaine différent ou identique à votre domaine DNS. Pour éviter les confusions, il est parfois préférable de distinguer ces 2 noms.

Pour ne pas avoir à le faire à chaque fois que vous redémarrez votre machine il suffit d'ajouter dans le fichier /etc/sysconfig/network la ligne

```
NIS_DOMAIN=mon_domaine
```

Le nom de domaine sera initialisé automatiquement lors du lancement de **ypserv**.

Aller ensuite dans le répertoire `/var/yp` et éditer le Makefile. Aller alors à la ligne `all:`, elle contient la liste des cartes possibles. Recopiez-là et commentez-là pour en garder une trace.

Laissez une ligne contenant simplement : `all:  passwd hosts group`

Ce sont les trois cartes généralement suffisantes pour la majorité des utilisations, en tout cas elles suffisent largement pour tester. Il sera ensuite toujours temps de rajouter les autres cartes dont vous avez besoin, comme par exemple la carte de `/etc/networks`.

Ensuite il n'y a plus qu'à faire un make, les cartes vont alors être créés dans le répertoire `/var/yp/mon_domaine`.

Il faut également éditer le fichier `ypserv.conf`

```
sunos_kludge: no
tryresolve: no
dns: no

# Host           : Map            : Security :Passwd_mangle
#
192.168.1.       : passwd.byname   : port     : yes
192.168.1.       : passwd.byuid    : port     : yes

# Not everybody should see shadow password, not secure, since
# under MSDOG everybody is root and can access ports < 1024 !!! *
                 : shadow.byname   : port     : yes
```

Enfin, il faut autoriser les clients à se connecter, pour cela il faut éditer le fichier hosts.allow si ce n'est pas encore fait pour qu'il ressemble à quelque chose comme cela :

```
ALL: 192.168.1.          ou bien
ALL: fresne, pantera
```

Il ne reste plus qu'à lancer le démon **ypserv**.

Le serveur NYS à proprement parler est prêt. Pour que les utilisateurs puissent se connecter de façon transparente sur les clients, il faut maintenant exporter le répertoire de leurs comptes par NFS.
Vous pouvez alors exporter les répertoires `/home/un_utilisateur` de chaque utilisateur par NFS et les monter sur le client, de préférence en lecture-écriture.

Toutefois, si vous avez un grand nombre d'utilisateurs avec NYS, il parait judicieux de les regrouper dans un seul et même répertoire, `/nisusers` par exemple que vous n'aurez plus qu'à exporter. Dans ce cas, il faudra également modifier le champ *home_directory* du fichier `/etc/passwd`, et le script **adduser** pour qu'il soit compatible.

11.2 Le client NYS

Il y a encore quelque temps, il était nécessaire de lancer un démon sur le client NIS. Ce client s'appelait *ypbind*, mais il est maintenant devenu inutile avec la distribution RedHat. En effet les fonctions de NIS ont été intégrées directement dans les librairies, qui ont maintenant le nom de NYS.

La documentation n'est malheureusement pas très claire à ce sujet, et il est parfois difficile de trouver la bonne façon de configurer son client NYS car les différentes docs que j'ai vues se mélangeaient allègrement les pieds entre les 2 implémentations.

Ce sont les fonctions NYS qui sont maintenant utilisés et dont nous parlerons dans ce document.

Il faut tout d'abord configurer comme pour le serveur le nom de domaine. Vous pouvez mettre dans le fichier `/etc/sysconfig/network` la ligne :

```
NIS_DOMAIN=mon_domaine
```

Cela me parait une bonne solution, mais comme le script **ypserv** n'est évidemment pas lancé, il faudra définir `domainname` dans un des scripts de lancement.

J'ai donc mis dans `/etc/rc.d/rc.sysinit` à la suite du hostname la ligne `domainname {$NIS_DOMAIN}`

```
   Il faut ensuite créer le fichier /etc/yp.conf
domainname mon_domaine_nis
ypserver mon_serveur_yp
ypsever  mon_deuxieme_serveur
```

Il permet d'indiquer le nom du serveur yp que l'on utilisera. Il est tout à fait possible de spécifier plusieurs serveurs avec la ligne *ypserver*. Celui qui sera utilisé est celui qui répondra en premier. Cela sert essentiellement pour des machines qui sont souvent déplacées entre des réseaux différents.

```
  Éditer également le fichier /etc/nsswitch.conf
passwd:     files nis
group:      files nis
hosts:      files nis dns
networks:   files nis
```

Ce fichier permet de déterminer par où commencera la recherche d'une information dans les différentes cartes.

Un exemple de ces 2 fichiers se trouve dans `/usr/doc/libc-5.3.12-8/`

À partir de là, les différentes fonctions de NYS doivent fonctionner.

```
  Pour le tester, essayons simplement un
$ypcat passwd
```

Généralement NYS est utilisé avec NFS pour offrir les même comptes utilisateurs sur toutes les machines. Pour cela, exporter les répertoires home des utilisateurs et distribuer les mots de passe par NYS. Ainsi, les utilisateurs peuvent se loguer sur n'importe quel client, et trouvent leur répertoire home et leurs fichiers de travail sur toutes les stations du réseau.

Concrètement, sur votre client une fois NYS correctement configuré, il suffit de monter les répertoires utilisateurs que le serveur a exporté. Si vous voulez l'automatiser, vous pouvez bien entendu mettre l'entrée dans votre `/etc/fstab`.

11.3 Les fichiers hosts.equiv et .rhosts

Lorsque l'on utilise NYS, on peut généralement utiliser son compte sur plusieurs clients. Pour des raisons pratiques on est parfois amené à se reloguer sur les autres stations régulièrement, et il devient parfois

agaçant de devoir sans cesse taper son mot de passe alors que l'on reste sur des clients locaux parfaitement authentifiés.

Il existe donc un moyen pour améliorer la souplesse du **rlogin**. Il suffit de définir dans le fichier hosts.equiv les machines et les comptes à partir desquels on autorise un rlogin sans mot de passe. Il suffit de mettre une ligne de la forme : `machine : bob, marcel`, pour que *bob* et *marcel* provenant de *machine* puissent se reloguer sans authentification. Les lignes contenant juste un nom de machine indiquent que tous les utilisateurs venant de cette machine peuvent se loguer sans authentification

Le fichier `.rhosts` qui existe dans le répertoire home des utilisateurs permet à ceux qui viennent des machines spécifiées de se connecter dans ce compte sans authentification.

Il est clair qu'il faut utiliser cette possibilité avec légèreté et ne pas autoriser n'importe qui (voir autoriser tout le monde) dans ces fichiers au risque de sérieusement réduire la sécurité de son site...

11.4 Le démon yppasswd

Lorsque l'on utilise NYS et les mots de passe distribués, la commande passwd sur un client risque de ne pas avoir le comportement attendu puisque qu'elle va éditer le fichier local `/etc/passwd`.

C'est donc le démon **yppasswd** du serveur qui doit se charger de cela. En fait lorsqu'un utilisateur voudra changer son mot de passe, il utilisera la commande **yppasswd**, qui ira modifier le fichier `/etc/passwd` du serveur NYS, et qui également mettra à jour les cartes, en faisant appel aux fonctions de notre bon démon.

Pour que l'utilisation de yppasswd soit transparente pour les utilisateurs, vous pouvez renommer le fichier `/usr/bin/passwd` en `lpasswd` par exemple, et ensuite vous faites un lien passwd vers yppasswd avec la commande :

```
ln -sf yppasswd passwd
```

Les utilisateurs pourront ainsi changer leur mot de passe sans se rendre compte qu'ils utilisent un compte NYS.

12 Samba : une porte vers les réseaux MS-Windows

Samba est une collections d'outils disponibles de façon libre sur les systèmes Un*x qui permettent d'accéder à un réseau MS-Windows. Samba permet ainsi à une machine Linux de devenir serveur de fichier pour tout un réseau sous MS-Windows, ou encore d'accéder aux différentes ressources qu'un serveur MS-Windows aura partagé.

12.1 Configuration de Samba sur la machine Linux

Si vous avez installé correctement samba, les 2 démons **smbd** et **nmbd** doivent êtres lancés automatiquement au boot de votre ordinateur. Si ça n'est pas le cas installez les packages avec *rpm* ou *glint*.

Les fichiers de configurations par défaut de samba se sont grandement améliorés récemment, en tout cas ceux fournis dans la RedHat m'ont permis d'accéder à mes répertoires homes assez facilement. Du moins, ils étaient suffisamment commentés pour permettre de servir de modèle à des modifications personnelles.

Cependant, je vais fournir une copie de celui de ma machine, qui permet de partager les répertoires home des utilisateurs, et offre un répertoire public pour tous les utilisateurs depuis la machine MS-Windows. De plus, il permet au serveur Samba (sous Linux) d'utiliser et de monter les ressources qui sont partagées par la machine MS-Windows.

```
[global]
   workgroup = ZEP_group
   comment = Tonio Samba Server
   volume = RedHat4

   printing = bsd
   printcap name = /etc/printcap
   load printers = yes

   log file = /var/log/samba-log.%m
   max log size = 50
   lock directory = /var/lock/samba
   locking = yes
   strict locking = no
   share modes = yes

   security = SHARE
   null passwords = yes
   socket options = TCP_NODELAY

; Permet au serveur Samba de devenir serveur du domaine
   os level = 33
   domain master = yes

; Facilite la gestion des noms longs
   preserve case = yes
   short preserve case = yes

; Chaque client accède à son répertoire
[homes]
   comment = Home Directories
   browseable = yes
   read only = no
   create mode = 0750

; Service d'impression pour partager les imprimantes
[printers]
   comment = All Printers
   path = /var/spool/samba
```

```
   browseable = no
   printable = yes
   public = no
   writable = no
   create mode = 0700

; Répertoire public /home/samba, accessible en lecture/écriture pour
; tout le monde
[public]
   comment = Repertoire Public
   path = /home/samba
   public = yes
   writable = yes
   browsable = yes
   printable = no
```

Une fois le fichier de configuration modifié, il faut relancer Samba avec :

```
/etc/rc.d/init.d/smb stop
/etc/rc.d/init.d/smb start
```

Enfin, si vous modifiez le fichier `smb.conf`, utilisez l'utilitaire **testparm** qui permet de vérifier la cohérence des informations de votre fichier.
Il permet également d'afficher la totalité de la configuration de votre serveur. Ce qui permet de vérifier la validité de vos modifications.

12.2 Configuration de la machine MS-Windows

Pour le moment, Samba ne permet d'accéder à la machine MS-Windows que si elle utilise une pile TCP/IP. En attendant la version qui devrait offrir un service direct avec NetBIOS. Il est donc nécessaire de configurer sous MS-Windows le support TCP/IP pour votre carte Ethernet, et d'affecter une adresse IP à votre machine.

Ceci étant fait après quelques reboots, devrait apparaître dans votre Voisinage Réseau votre nouveau serveur Samba sous Linux. Soyez patients, le support réseau de MS-Windows met parfois plus d'une minute pour "voir" les autres ordinateurs. Cependant en faisant *Démarrer -¿ Rechercher Ordinateur* et en entrant le nom de votre serveur Samba, il devrait s'afficher de manière quasi-instantanée.

Vous allez normalement alors pouvoir accéder au compte de l'utilisateur avec lequel vous êtes connectés sous MS-Windows, ainsi qu'à notre répertoire public */home/samba*

12.3 Utilisations des outils smbclient et smbmount sur la machine Linux

Smbclient et smbmount permettent d'accéder aux différentes ressources offertes par le serveur MS-Windows.

12.3.1 smbclient : le ftp pour MS-Windows

smbclient permet de tester, répertorier et accéder aux ressources offertes par le serveur MS-Windows.

Tout d'abord, il est possible de vérifier si le serveur MS-Windows est bien disponible, et de lister les différentes ressources qu'il partage grâce à la commande :

```
$smbclient -L nom_serveur_windows
```

nom_serveur_windows représentant le nom que vous avez donné à votre machine sous MS-Windows.

Ensuite, il est possible d'accéder au différents fichiers de la ressource partagée en s'y connectant avec **smbclient**, puis de la parcourir et de récupérer les fichier exactement comme avec un ftp :

```
$smbclient \\\\nom_serveur_windows\\nom_ressource
```

Avec la majorité des Shell, il est nécessaire de doubler les backslash (\) pour qu'ils soient pris en compte. *nom_ressource* représente le nom de la ressource partagé que l'on peut obtenir avec un **smbclient -L**

Après cela, il devrait vous être demandé un mot de passe (qui correspond à celui spécifié pour la ressource sous MS-Windows). Une fois le mot de passe entré, il n'y a plus qu'à parcourir les fichiers exactement comme avec ftp.

12.3.2 Smbmount : monter sous Linux des répertoires MS-Windows

Smbmount permet de monter (comme par NFS) des répertoires MS-Windows sur l'arborescence Linux et de les manipuler le plus naturellement du monde.

```
$smbmount //nom_serveur_windows/nom_ressource /mnt -f 777
```

smbmount utilise les '/' à la différence de smbclient.

Il est nécessaire de rajouter une entrée dans le fichier `/etc/hosts` pour *nom_serveur_windows* avec son adresse IP, pour que smbmount puisse marcher correctement.

Enfin, il est nécessaire de passer quelques paramètres pour assurer la compatibilité des droits de MS-Windows avec les droits Un*x. La clause **-f 777** met tous les fichiers en lecture-écriture pour tout le monde.
On peut bien sûr la combiner avec l'option **-u uid** et **-g gid** qui permettrons de définir les droits précis des fichiers de la ressource pour pallier les carences de MS-Windows dans ce domaine.

13 APACHE / httpd : un serveur Web

Le démon httpd fourni avec la Redhat est celui écrit par Apache. Celui-ci permet à votre machine de devenir un serveur Web (aussi appelé WWW).

Sa configuration est des plus simple. Tout d'abord, il faut installer apache sur votre machine si ça n'est pas encore fait. À vous de choisir votre solution préférée : un bon *rpm* en ligne ou la version graphique *glint*

(qui ne présente pas toujours correctement les messages d'erreur éventuellement engendrés lors d'une installation avortée).

13.1 Configuration d'Apache

Tous les fichiers de configuration de httpd se trouvent dans le répertoire `/etc/httpd/conf/`. Le fichier principal est le fichier `httpd.conf`. Il suffit de réactiver la ligne ServerName qui est normalement commentée, et de spécifier le nom auquel le serveur répondra, par exemple :

```
ServerName nom_de_ma_machine
```

Bien évidemment, ce nom doit être un nom valide de la machine, c'est-à-dire un nom auquel elle répondra soit directement, soit après une résolution par NYS ou DNS. Pour le moment, mettre tout simplement le nom habituel de la machine.

Il ne reste plus qu'à tester le bon fonctionnement du serveur Web, en lançant un browser quelconque à l'adresse `http://nom_de_ma_machine/` ou `http://localhost/` si vous êtes directement sur le serveur. Cela devrait afficher la page de présentation de apache. Celle-ci est installée dans le répertoire `/home/httpd/` où vous pourrez bien entendu placer vos pages Web.

Les utilisateurs peuvent maintenant insérer leurs pages Web dans un répertoire *public_html* sur leur compte qui sera accessible par `http://nom_de_ma_machine/~user_name/`. Le nom de ce répertoire ainsi que les pages chargées par défaut (style index.html) sont paramètrés et modifiables dans le fichier `srm.conf`.

13.2 Accès aux fichiers et gestion des droits d'accès

Avant tout, pour que le démon httpd puisse accéder aux fichiers, ceux-ci doivent avoir le droit de lecture pour tous.

Pour l'appliquer à tous les fichiers d'un répertoire, nous lancerons ainsi la commande `chmod a+r *`.

Par ailleurs, Apache fournit un système permettant de définir les droits d'accès des différents répertoires. Les droits d'accès par défaut sont définis dans le fichier `/etc/httpd/conf/access.conf`.

Il est toutefois possible de préciser un droit d'accès pour chaque répertoire, ce qui peut être utile si l'on installe des pages qui doivent être accessibles uniquement de façon interne à votre entreprise/association/réseau, et non à tous les internautes.

Cela se fait dans un fichier appelé par défaut `.htaccess` qui contient par exemple :

```
order deny,allow
deny from all
allow from .votre_domaine.fr
```

Ce qui permet d'autoriser uniquement les utilisateurs de `.votre_domaine.fr` à accéder aux fichiers du répertoire où est placé ce fichier.

13.3 Configuration et utilisation des scripts CGI

L'un des principaux intérêts d'un serveur Web est de permettre d'exécuter des programmes sur le serveur. Ces programmes sont généralement dénommés des scripts cgi.

Apache permet bien évidemment d'exécuter de tels scripts. Par défaut, ces scripts sont autorisés à être exécuté uniquement si ils sont stockés dans le répertoire : `/home/httpd/cgi-bin`.

Mettons donc dans ce répertoire un petit script perl `test.pl` :

```
#! /usr/bin/perl

print "Content-type: text/html\n\n";
print "Tout baigne\n";
```

Et à l'affichage par votre browser préféré de `http://nom_de_ma_machine/cgi-bin/test.pl` vous lirez le message "Tout baigne".

Vous pouvez également autoriser le lancement de scripts CGI dans d'autres répertoires ajoutant l'option `AddHandler cgi-script .cgi` dans votre fichier `srm.conf`. Cependant, il est préférable de limiter au cas par cas cette possibilité pour des raisons de sécurité grâce au fichier `access.conf`.

La configuration fournie ici est suffisante pour la majorité des utilisations, mais les fichiers de configuration sont suffisamment bien commentés pour ne pas mériter de plus amples détails. Il ne reste plus qu'à vous lancer et configurer des VirtualHosts, le proxy cache, le support java et les trucs sympathiques du genre...

14 PPP : relier son PC Linux à Internet

Le protocole PPP (Point to Point Protocol) permet entre autre de relier votre machine à Internet en utilisant votre modem. Il permet ensuite d'utiliser tous les services IP, avec toutefois quelques restrictions étant donné que généralement votre adresse IP sur Internet sera dynamique.

14.1 Configurations préalables (modem, port série, DNS)

Une fois vérifié que **pppd** et **chat** sont bien installés sur votre machine, il ne reste plus qu'à se connecter en ayant pris la peine de récupérer quelques informations indispensables.

- Le port série sur lequel est branché votre modem
- Le numéro d'appel de votre fournisseur d'accès Internet (FAI)
- Le nom utilisateur et le mot de passe de votre FAI
- Et l'adresse du serveur DNS de votre FAI

Les ports série sous Linux ont la dénomination **ttySx**, ou x correspond au numéro du port série. Sachant qu'il commence à 0. Le port DOS COM1 est donc ttyS0, COM2 est ttyS1, et ainsi de suite...

Pour le modem, dans le répertoire **dev**, il faut créer ou modifier le lien `/dev/modem` pour qu'il pointe sur le port correspondant au modem de votre machine. Par exemple (pour un modem sur le COM4) :

```
ln -sf /dev/ttyS3 /dev/modem
```

L'adresse du serveur DNS est parfois un petit peu plus délicate à obtenir car elle n'est pas indispensable pour la configuration de PPP avec MS-Windows. Cette information est cependant indispensable pour la configuration de PPP sous Linux. Il faut donc parfois insister poliment auprès de FAI pour l'obtenir.

Il faut ensuite ajouter cette adresse DNS dans le répertoire `/etc/resolv.conf` :

```
domain mon-domaine-internet.fr
nameserver 194.250.238.1
nameserver 190.158.97.67
```

`mon-domaine-internet.fr` est à remplacer par le nom de domaine correspondant à votre FAI.

Les adresses IP suivant la clause **nameserver** sont à remplacer par les adresses IP du DNS fourni par votre FAI. Il est préférable de mettre deux lignes nameserver pour avoir une solution de remplacement au cas ou le premier serveur de noms (DNS) tomberait en panne.

Le fichier `/etc/hosts.conf` doit être de la forme

```
order hosts,bind
multi on
```

Remarque : Il faut généralement être **root** pour pouvoir lancer une connexion PPP, car `pppd` manipule différentes informations des tables de routages et des interfaces qui nécessitent d'avoir ces droits.

14.2 Scripts de connexion

Il faut ajouter le script pour lancer la connexion. Je propose de le placer dans `/usr/local/bin/` et de l'appeler **startppp**.

```
#/bin/sh

/usr/sbin/pppd connect '/usr/sbin/chat -v ABORT ERROR ABORT "NO CARRIER" \
ABORT BUSY  "" ATZ OK ATDT0146402992 CONNECT "" ogin: "MON_LOGIN" \
word: "MON_MOT_DE_PASSE"' \
/dev/modem 38400 noipdefault debug crtscts modem defaultroute &
```

Les options à modifier selon sa configuration :

- **ATDT0146402992** : à remplacer par le numéro de téléphone de votre FAI qui va bien.
- **MON_LOGIN** : à remplacer par votre le login chez votre FAI
- **MON_MOT_DE_PASSE** : votre mot de passe chez votre FAI

- **/dev/modem** : si le périphérique /dev/modem n'est pas défini ou si vous préférez, vous pouvez mettre le périphérique réel /dev/ttyS3 pour le COM 4 par exemple.
- **38400** : c'est le débit entre votre PC et votre modem, si vous avez un modem très rapide mieux l'adapter. Prenez alors le maximum : 115200.

De façon identique, nous allons créer un script qui permet de terminer la connexion que je propose de mettre lui-aussi dans `/usr/local/bin` sous le nom **stopppp**:

```
#!/bin/sh
killall -9 pppd
```

À partir de là, si tout va bien et que vous n'avez rien oublié, tout devrait fonctionner correctement. Si tel est le cas, vous pourrez enlever les options *-v* de `chat` et *debug* de pppd pour économiser les messages de logs qui prennent de la place inutile lorsque tout fonctionne correctement.

Pour vérifier lancer la commande **startppp** ou le nom du fichier que vous lui avez donné (en étant root). Au bout de quelques secondes, la commande **ifconfig** devrait indiquer la présence d'un nouveau périphérique **ppp**.

Il n'y a plus qu'à essayer un **ping** de l'adresse du serveur DNS ou d'une machine existante sur Internet pour s'en assurer. À partir de là lancer votre navigateur préféré (par exemple Lynx) et partez à l'aventure, il devrait fonctionner...

Si malgré tout vous ne parvenez pas à faire fonctionner votre connexion, jettez un coup d'oeil au PPP-HOWTO qui couvre un domaine plus large, ou encore consultez *http://www.linux-france.com/article/connex/* `<http://www.linux-france.com/article/connex/>` qui regroupe de nombreuses configurations de PPP pour les principaux FAI français.

14.3 Consulter les News et le Mail chez son FAI

Une fois que la connexion PPP fonctionne, Netscape configuré correctement doit vous permettre de consulter le Web, lire les news, retirer et envoyer votre mail chez votre FAI.

Mais je vois d'ici que bon utilisateur GNU que vous êtes, cette solution ne vous convient qu'à moitié !

14.3.1 News Usenet

Tout bon lecteur de news qui se respecte (slrn, trn ...) doit pouvoir utiliser la variable d'environnement *NNTPSERVER* pour définir le nom du serveur de news.

Il suffit donc avant de lancer votre lecteur de news favori de définir cette variable avec la commande :

```
export NNTPSERVER=news.provideur.fr
```

On peut aussi le définir dans le fichier de configuration du lecteur de news (consulter éventuellement le man de celui-ci).

Remarque : news.provideur.fr est bien entendu à remplacer par l'adresse exacte de votre serveur de news.

Un document décrit par ailleurs *divers outils de lecture hors ligne*
<http://www.linux-france.com/article/usenet/>.

14.3.2 mail

Consulter les documents *http://www.linux-france.com/article/mail/*
<http://www.linux-france.com/article/mail/>

15 NFS-Root : booter une machine par le réseau

NFS-Root permet d'utiliser une partition NFS comme partition root (/). Cette possibilité est particulièrement utile pour une station dépourvue de disque local (diskless), un terminal X, un serveur d'impression ou un ordinateur ne disposant pas de disque dur dédié à Linux.

Le principe est de créer sur un serveur l'arborescence du client. Celui-ci bootera à l'aide d'un noyau sur une disquette ou sur une partition MS-DOS avec LOADLIN, puis utilisera NFS pour récupérer les fichiers nécessaires à la suite de son démarrage.

15.1 Création de l'arborescence du client

Il faut créer sur le serveur toute l'arborescence nécessaire au fonctionnement du client. Une solution consiste à recopier froidement la racine de son serveur puis de modifier les fichiers spécifiques. Cette solution est extrêmement gourmande en place surtout si l'on envisage plusieurs clients utilisant NFS-Root. Je propose donc une solution qui permet de limiter le nombre de fichiers à recopier en utilisant directement ceux disponibles sur le serveur.

15.1.1 Création de la partition root du client

Nous allons essayer de limiter au maximum la taille de ce répertoire en utilisant des liens symboliques et en recopiant uniquement les fichiers indispensables au boot ainsi que ceux spécifiques à chaque station. Les autres répertoires étant soit directement ceux du serveur exportés par NFS, soit un répertoire commun aux différents clients, différent du serveur et également exporté par NFS.

Par défaut NFS-Root monte le répertoire `/tftpboot/client-IP`.
client-IP étant l'adresse IP du client qui aura soit été passé en paramètre lors du boot, soit découverte par une requête rarp. Ce chemin peut être changé à la compilation du noyau ou passé en paramètre à LILO.

Dans ce répertoire `/tftpboot/client-IP`, il faut donc créer les répertoires :

```
bin, dev, etc, home, lib, mnt, proc, sbin, commun, tmp, usr, var
```

ainsi que les autres répertoires dont vous avez besoin.

Il faut maintenant remplir les différents répertoires :

- **bin** doit contenir les programmes **mount, sh** *(qui est généralement un lien vers* **bash** qu'il faut donc également copier, sinon recopier le shell pointé par sh)
- **dev** doit contenir l'ensemble des accès aux périphériques. Pour le générer, utiliser :

  ```
  cp -a /dev /tftpboot/client-IP/dev
  ```

  ne pas oublier que les devices *console, mouse, cdrom* et *modem* sont des liens symboliques, à modifier le cas échéant.
- **sbin** doit contenir le programme **init**
- **lib** doit avoir quelques librairies indispensables, à recopier avec :

  ```
  cp -a /lib/libc.so* /lib/libtermcap.so* /lib/ld* /tftpboot/client-IP/lib
  ```

Pour ces quatre répertoires, des liens "durs" peuvent également être utilisés au lieu de recopier les fichiers, permettant d'économiser autant de place disque, mais avec les soucis que peuvent entraîner les liens durs... A réserver donc aux utilisateurs expérimentés.

15.1.2 Les répertoires var et etc, partagés entre les clients

Le répertoire **var** doit avoir une partie privée, et une partie commune aux autres clients.
Le répertoire commun à tous les clients, sera placé dans /tftpboot/commun/. Il contiendra une copie des répertoires **/var/catman** et **/var/lib** du serveur, qui sont particulièrement volumineux.

Le répertoire `/tftpboot/client-IP/var`, contiendra donc tous les répertoires restant de var. En recopiant celui-ci à partir du serveur, veillez à bien conserver l'arborescence, mais n'hésitez pas à détruire les fichiers de logs créés et qui augmentent à chaque démarrage. On créera aussi 2 liens symboliques :

```
ln -s /commun/catman /tftpboot/client-IP/var/catman
ln -s /commun/lib    /tftpboot/client-IP/var/lib
```

Le même principe sera utilisé pour le fichier **termcap** de **etc**, relativement volumineux. Il sera recopié également dans le répertoire /tftpboot/commun/, avant de faire le lien :

```
ln -s /commun/termcap /tftpboot/client-IP/etc/termcap
```

Les autres fichiers de **etc** nécessaires au client seront enfin recopiés dans le répertoire `/tftpboot/client-IP/etc`

Avec cette méthode, le répertoire root de chaque client fait, chez moi, 1.4 Mo, ce qui reste très résonnable comparé à la taille que peut prendre une copie de l'intégralité d'une arborescence !

15.1.3 Paramétrage du client, avec les fichiers de etc

Il est enfin nécessaire de configurer les différents fichiers de `/tftpboot/client-IP/etc/` qui vont permettre à notre client de booter correctement par NFS.

Tout d'abord, le fichier **fstab** indispensable pour monter les différentes partitions doit contenir au moins :

```
/dev/root                   /             nfs     defaults        1 1
serveur:/bin                /bin          nfs     defaults        1 1
serveur:/usr                /usr          nfs     defaults        1 1
serveur:/sbin               /sbin         nfs     defaults        1 1
serveur:/home               /home         nfs     defaults        1 1
serveur:/lib                /lib          nfs     defaults        1 1
serveur:/users              /users        nfs     defaults        1 1
serveur:/tftpboot/commun    /commun       nfs     defaults        1 1
none                        /proc         proc    defaults        0 0
```

serveur étant bien entendu à remplacer par le nom du serveur NFS-Root.

Il faut également modifier le *nom de la machine*, le fichiers *hosts* et les différents fichiers de la machine décris dans les sections précédentes.

Dans le fichier `rc.d/rc.sysinit`, rajouter au début les 2 lignes :

```
mount -a
echo "NFS-Root" > /fastboot
```

Cela permet de monter le reste de l'arborescence. Le fichier `/fastboot` permet d'éviter les fsck inutile pour les partitions NFS. Il est également nécessaire de commenter ou de supprimer toutes les autres lignes faisant un `mount` ou un `umount`.

Dans les différents répertoires *rcn.d*, supprimez les services inutiles. De plus il faut supprimer dans rc.d/rc3.d/ les liens `S10network et S15nfsfs` qui génèrent des messages d'erreur, et font même planter le système lorsqu'ils tentent de reconfigurer les interfaces réseau alors déjà utilisées.

Enfin je vous conseille de créer des fichiers `passwd` et `group` minimaux. Utiliser NYS pour le client parait une solution naturelle et efficace pour conserver ces fichiers à jour, à plus forte raison dans le cas ou plusieurs clients NFS-Root sont envisagés.

15.2 Configuration du serveur

La configuration du serveur est relativement rapide. Tout d'abord, ajouter une entrée pour votre client dans le fichier `/etc/hosts`, modifier le fichier `/etc/hosts.allow` également pour que le client puisse accéder aux partitions exportés par NFS.

Enfin, ajouter au fichier `/etc/exports` :

```
/tftpboot/192.168.1.120         kashmir(rw,no_root_squash)
/tftpboot/commun                kashmir(ro,no_root_squash)
```

```
/usr                    kashmir(ro,no_root_squash)
/sbin                   kashmir(ro,no_root_squash)
/bin                    kashmir(ro,no_root_squash)
/lib                    kashmir(ro,no_root_squash)
/home                   kashmir(rw,no_root_squash)
/users                  kashmir(rw,no_root_squash)
```

`kashmir` étant le nom de mon client correspondant à l'entrée de */etc/hosts* qui est comme vous l'avez remarqué doté de l'adresse 192.168.1.120

Ne pas oublier ensuite de redémarrer le service NFS.

Le noyau du serveur doit être également configuré avec le support *NFS* et *Reverse ARP*. Pour s'en assurer, vérifier l'existence du fichier `/proc/net/rarp` et de la ligne `nodev nfs` dans `/proc/filesystems`

Enfin configurer rarp permet au client de booter de façon automatique. Pour cela, ajouter la ligne :

```
/sbin/rarp -s 192.168.1.120 00:40:F6:7A:BC:45
```

le premier paramètre étant l'adresse IP du client, et le second étant l'adresse de la carte Ethernet.

Pour connaître l'adresse de votre carte Ethernet, bootez sur votre client un noyau avec le support de votre carte Ethernet. Les 6 chiffres affichés à ce moment là doivent correspondre à ce que vous cherchez.
Il peut être judicieux d'inclure cette ligne dans `/etc/rc.d/rc.sysconfig` pour ne pas avoir à taper la commande à chaque boot du serveur.

15.3 Création du noyau et paramétrage du client lors du boot

Le noyau du client doit avoir les options NFS-Root et Rarp configurées. De plus n'oubliez pas la configuration de votre carte Ethernet, et éventuellement des périphériques SCSI ou autres.

Si vous comptez démarrer votre station sans entrer de paramètres, il faut :

- Configurer Rarp sur le serveur pour votre client.
- Créer un device virtuel en faisant : `mknod /dev/nfsroot b 0 255`
- Puis faire sur le fichier image du noyau un : `rdev (kernel-image) /dev/nfsroot`
- Enfin le recopier sur la disquette de boot avec un simple : `cp (kernel-image) /dev/fd0`

Si tout va bien, la station devrait démarrer correctement.

Si toutefois, la configuration de rarp pose des problèmes, ou que le client ne se trouve pas sur le même noeud que le serveur, vous pouvez passer des paramètres au noyau pour contourner ces soucis. Ils peuvent même éventuellement être ajoutés dans la close *append* du `lilo.conf`

Ainsi vous pouvez modifier le chemin par défaut `/tftpboot/client-IP` en passant au noyau le paramètre :

```
nfsroot=serveur-IP:/chemin/root-dir
```

`serveur-IP` étant un paramètre facultatif permettant de donner l'adresse du serveur NFS-Root éventuellement différent du serveur Rarp.
`/chemin/root-dir` étant le nouveau chemin du répertoire contenant l'arborescence root pour le client. Un token "%s" sera remplacé par la représentation ASCII de l'adresse IP du client.
L'adresse par défaut étant donc noté "/tftpboot/%s"

Un autre paramètre permet de donner au client toutes les informations nécessaires à son démarrage, avec la ligne suivante :

```
nfsaddrs=(client-IP):(serveur-IP):(gw-IP)::(hostname)::none
```

- *(client-IP)* étant l'adresse IP du client
- *(serveur-IP)* étant l'adresse IP du serveur
- *(gw-IP)* IP du gateway si le serveur est sur un autre sous-réseau(facultatif).
- *(hostname)* étant le nom donné au client

En passant ces paramètres au noyau, la configuration de rarp est inutile et ignoré. Cela peut-être pratique si le client NFS-Root est lancé sur différentes machine. En effet il n'y aura alors pas besoin de se soucier de l'adresse de la carte Ethernet.

Il est également possible d'utiliser **bootp** ou encore **dhcpd** qui sont plus complets Rarp pour que le client récupère ses paramètres, mais leur configuration est un peu plus délicate et de toute façon, rarp suffit largement à la majorité des utilisations.

15.4 Ajouter un nouveau client

Une fois l'arborescence du premier client créée et fonctionnant, la mise en place de plusieurs autres clients est une partie de plaisir.

Il suffit de recopier le répertoire root `/tftpboot/client-IP` du premier client dans un nouveau répertoire :

```
cp -a /tftpboot/client-IP /tftpboot/nouveau-client-IP
```

nouveau-client-IP étant l'adresse IP du nouveau client.

Ensuite il suffit de modifier les fichiers désignant le nom du nouveau client `etc/HOSTNAME`, `etc/hosts` et `etc/sysconfig/network`, et éventuellement d'autres fichiers de `etc` si le nouveau client à de nouvelles spécificités.

Sur le serveur, enfin, il faut ajouter l'adresse de notre nouveau client au fichier `/etc/hosts`. Ajouter une entrée dans `/etc/exports` pour le répertoire root, et lui donner les droits pour tous les autres répertoires qu'il partage avec les différents clients *(/tftpboot/commun /bin /usr /lib /sbin /home /users)*.
Chaque ligne ressemble pour 2 client (*kashmir* et *firestarter*) à :

```
/lib              kashmir(ro,no_root_squash) firestarter(ro,no_root_squash)
```

Relancer enfin le service NFS.

Les 2 clients sont maintenant capable de démarrer et de fonctionner ensemble sur le serveur.