# NFS-Root client et serveur HowTo

Hans de Goede <hans@highrise.nl> v1.0 30 Mars 1999

Ce document décrit l'installation et la configuration d'un serveur pour que ses clients puissent démarrer et fonctionner sans disque (par montage NFS de root).

## Contents

# 1 Introduction

Ce Howto est également disponible à l'adresse -

`http://x.mame.net/hans` . Il décrit un exemple de configuration où root est monté en NFS, mais il diffère des autres NFS-root HowTo sur deux points :

1. Il propose à la fois l'aspect serveur et l'aspect client, offrant une solution complète ; il ne décrit pas les principes de base du montage de root via NFS mais plutôt un exemple de configuration qui fonctionne.
2. La configuration décrite est particulière dans la mesure où c'est l'arborescence root du serveur qui est partagée avec les stations (workstations, ws), tandis qu'habituellement, on a plutôt un mini-root par station. Ceci a quelques avantages :
   - occupe un faible espace disque
   - tous les changements sur le serveur sont automatiquement disponibles côté client (la configuration n'est faite qu'une fois)
   - facilite l'ajout de nouveaux clients
   - un seul système à maintenir.

Ce document est basé sur un système RedHat 5.2. On suppose que le lecteur de ce HowTo a suffisamment d'expérience en administration système linux ; l'adaptation de cette solution à une autre distribution ne devrait donc pas poser de problème.

## 1.1 Copyright

## 1.2 Historique

- v0.1, 20 Janvier 1999 : premier jet au HHS, là où la configuration a été développée.
- v1.0, 30 Mars 1999 : première version diffusée, écrite partiellement durant ma période de stage chez ISM.

# 2 Principes de base

Dans cette configuration les clients utilisent le système de fichiers racine du serveur. Ils y accèdent bien sûr en lecture seule.

## 2.1 Les choses ne peuvent pas être aussi simples...

Quelques problèmes apparaissent rapidement.

### 2.1.1 Chaque station a besoin de sa propre copie d'un certain nombre de répertoires

Une configuration linux doit avoir les accès en écriture sur les répertoires suivants :

1. /dev
2. /var
3. /tmp

Il y a trois solutions, l'une d'elles ne fonctionnant que pour /dev :

1. utiliser (monter) un ramdisk et remplir celui-ci par extraction d'une archive ou copie depuis un répertoire modèle.
   - avantages :
     (a) nettoyé à chaque reboot (suppression des fichiers tmp et log). Pas de maintenance.
     (b) ne prend pas de place sur le serveur et ne génère pas de trafic réseau. Il est donc plus rapide et utilise moins de ressources côté serveur.
   - inconvénients :

   (a) occupe de la mémoire
   (b) les fichiers de log ne sont pas conservés. Il faut configurer syslog pour rediriger les logs sur le serveur si on tient vraiment à récupérer les messages des clients.

2. créer un répertoire pour chaque station sur le serveur et le monter par NFS en lecture-écriture.

   - avantages & inconvénients :

     (a) les arguments ci-dessus sont à prendre à l'envers dans le cas des répertoires situés sur le serveur.

3. à partir du noyau 2.2, on peut utiliser le type devfs pour /dev (un système de fichiers virtuel à la manière de /proc).

   - avantages:

     (a) devfs prend très peu de mémoire comparé à un ramdisk, et pas du tout d'espace disque sur le serveur. En plus il est très rapide. Un /dev normal occupe au moins 1,5 Mo dans la mesure où un fichier (un *device*) fait au minimum 1 ko, et il y a environ 1200 fichiers. On peut bien entendu utiliser un modèle de /dev avec simplement les entrées nécessaires pour économiser un peu de place : 1,5 Mo, ça fait beaucoup pour un ramdisk et ça ne fait pas très propre sur un serveur.
     (b) devfs crée automatiquement des entrées pour les devices détectés et ajoutés, donc pas besoin de maintenance.

   - inconvénients :

     (a) tout changement sur /dev tel que création d'un lien pour la souris ou le lecteur de cdrom est perdu. Devfs fournit cependant un script nommé rc.devfs pour sauvegarder ces changements. Le script présent dans ce HowTo va alors automatiquement restaurer les liens symboliques nouvellement positionnés en appelant rc.devfs. Si on fait des changements sur /dev, il faut donc appeler rc.devfs soi-même de cette façon :

         /etc/rc.d/rc.devfs save /etc/sysconfig

Comme on peut le voir, il y a plusieurs moyens de résoudre ce problème d'accès en lecture-écriture. Voici les options choisies pour le reste de ce Howto :

- pour /dev nous utiliserons devfs
- pour /var et /tmp, nous utiliserons un ramdisk de 1 Mo. Celui-ci sera partagé pour utiliser la mémoire de manière efficace. Pour réaliser ce partage, /tmp sera en fait un lien symbolique sur /var/tmp.
- pour remplir ce ramdisk, une archive conviendra tout aussi bien qu'un répertoire modèle. Mais comme les modifications sont plus aisées avec le répertoire modèle, c'est cette dernière solution qui sera retenue.

### 2.1.2 Un accès en écriture sur /home semble nécessaire...

Mais ce n'est pas vraiment un problème puisque dans toute configuration unix de type client/serveur, /home est monté en lecture-écriture depuis le serveur, donc ça nous conviendra ;)

### 2.1.3 Comment une station récupère son adresse IP de manière à pouvoir communiquer avec le serveur ?

Heureusement pour nous ce probleme a déjà été résolu et le noyau a deux possibilités pour la configuration automatique de l'adresse IP :

1. RARP
2. Bootp

RARP est le plus facile à configurer, bootp est le plus flexible. Mais la plupart des bootroms supportent uniquement bootp, donc nous utiliserons bootp.

#### 2.1.4 Et la configuration spécifique à chaque station ?

Sur RedHat, la plupart des fichiers de configuration système sont déjà situés sous /etc/sysconfig. Nous déplacerons donc simplement ceux qui ne le sont pas encore et ajouterons des liens symboliques. Ensuite nous monterons un répertoire /etc/sysconfig par station. C'est la seule partie qui est propre à la distribution utilisée ici. Avec une autre distribution, il suffira de créer un répertoire sysconfig, déplacer tous les fichiers de configuration qui ne peuvent être partagés, et ajouter les liens nécessaires. De même, /etc/rc.d/rc3.d (ou l'équivalent dans les autres distribs) peut présenter des différences entre le serveur et les stations. Si on considère que toutes les stations lancent les mêmes services, on créera simplement un rc3.d pour les stations et un pour le serveur :

1. créer un /etc/rc.d/rc3.ws et un /etc/rc.d/rc3.server
2. faire un lien de /etc/rc.d/rc3.d vers /etc/sysconfig/rc3.d
3. faire un lien de /etc/sysconfig/rc3.d vers /etc/rc.d/rc3.xxx
4. remplacer S99local dans rc3.ws par un lien vers /etc/sysconfig/rc.local pour que chaque station ait son propre rc.local

#### 2.1.5 Divers problèmes

1. /etc/rc.d/rc.sysinit a besoin de /var, donc /var doit être monté ou créé avant que rc.sysinit ne soit exécuté. Il serait également intéressant que /etc/sysconfig (propre à chaque station) soit monté avant le lancement des scripts d'initialisation.
   - pour cela nous appellerons un script dès le début de /etc/rc.d/rc.sysinit, aussi bien sur le serveur que sur les stations ; ce script devra donc détecter sur quelle machine il tourne pour ne rien faire dans le cas du serveur.
2. /etc/mtab doit être accessible en écriture :
   - il suffit de créer un lien vers /proc/mounts et un fichier vide mounts dans /proc pour que fsck et mount ne se plaignent pas pendant l'initialisation (alors que /proc n'est pas encore monté). Il est à noter que smb(u)mount ne respecte pas le lien mtab et va l'écraser. Donc si on utilise smb(u)mount, il faut écrire un wrapper qui va restorer le lien.

# 3 Préparation du serveur

## 3.1 Compiler un noyau

Il faut prévoir le nécessaire pour supporter root sur nfs. Voici les étapes :

1. Comme nous utilisons une RedHat 5.2 avec le noyau 2.2, il faut s'assurer que notre distribution est prête pour ce noyau. RedHat fournit un excellent HowTo à ce sujet.

2. J'utilise le même noyau pour le serveur et les stations pour éviter les conflits vu qu'ils partagent le même répertoire /lib/modules. Si ce n'est pas possible dans votre situation, produisez différentes versions en éditant le numéro de version au début du Makefile. Ces numéros différents devraient éviter les confilts.

3. En plus des options habituelles, le noyau devrait supporter :

   - ext2 compilé dans le noyau (pour le serveur, ou bien pour les deux)
   - NFS et root-over-NFS compilé (pour le client ou pour les deux) ; pour avoir l'option root-over-NFS, il faut activer ip-autoconfig dans les options réseau. Nous utiliserons bootp comme méthode de configuration.
   - networkcard compilé (pour le client ou les deux)
   - devfs compilé (requis pour le client, également intéressant pour le serveur)
   - tout ce que vous utilisez normalement, les modules pour tous les périphériques présents sur le serveur et les stations.

4. Il faut éditer ensuite les sources du noyau pour changer le montage root-over-NFS par défaut : /tftpboot/<ip>/root au lieu de /tftpboot/<ip>, de façon à avoir une arborescence propre sous /tftpboot avec un répertoire par station contenant son répertoire racine (un lien vers la racine du serveur en fait) et ses répertoires spécifiques.

   - En 2.0, c'est une ligne de DEFINE dans "include/linux/nfs_fs.h" appelée "NFS_ROOT"
   - En 2.2, c'est un DEFINE dans "fs/nfs/nfsroot.c"

5. Il reste à compiler le noyau comme d'habitude (cf Kernel-HowTo).

6. Si vous n'avez pas encore de noeud /dev/nfsroot, créez-le :

   mknod /dev/nfsroot b 0 255

7. Après avoir compilé le noyau, changez la racine en tapant :

   rdev <path-to-zImage>/zImage /dev/nfsroot

8. Avant de booter avec devfs, vous devez modifier conf.modules : ajoutez le contenu du fichier conf.modules de la documentation de devfs au conf.modules du système.

9. Ce nouveau noyau est compilé avec la configuration automatique de l'adresse IP, mais cela va échouer lors du boot du serveur puisque c'est lui-même qui donne les adresses IP. Pour éviter une trop longue attente, ajouter : append="ip=off" à la section linux de /etc/lilo.conf.

10. relancez lilo et bootez sur le nouveau noyau.

11. avec devfs, sur le serveur, vous allez perdre tous les liens qui existaient. Sur RedHat, c'est le plus souvent /dev/mouse et /dev/cdrom. Recréez-les. Remettez également vos propriétés personnalisées si vous avez l'habitude d'avoir des particularités sur certaines entrées de /dev. Ensuite enregistrez ce paramétrage de /dev (sous /etc/sysconfig puisque c'est dépendant du type de machine) ainsi :

    - Copiez le fichier rc.devfs de la documentation devfs des sources du noyau vers /etc/rc.d/rc.devfs et rendez-le exécutable
    - Sauvegardez les paramétrages :

      /etc/rc.d/rc.devfs save /etc/sysconfig

## 3.2 Création et remplissage de /tftpboot, création des liens vers /tmp etc.

### 3.2.1 La partie automagique

Tout cela est pris en charge par le script ci-dessous. Si on veut le faire manuellement, il suffit de suivre le script pas a pas.

Ce script effectue des actions un peu osées telles que supprimer /tmp, arrêter temporairement syslog, démonter /proc. Donc assurez-vous d'abord que personne n'utilise la machine pendant ce temps, et que X ne tourne pas. Il n'est pas nécessaire de changer de niveau d'exécution, si vous êtes sûr d'être le seul connecté et sur une console en mode texte.

Déni : ce script a été testé mais s'il provoque un plantage du serveur, vous êtes seul responsable. Je ne prends aucune responsabilité quoi qu'il arrive. Je répète que ce HowTo est fait pour des administrateurs expérimentés. De plus ce script est fait pour être lancé une fois et une seule. Le lancer une seconde fois endommagera /etc/fstab, /etc/X11/XF86Config, /etc/X11/X et /etc/conf.modules.

Ceci dit, copiez-collez ce script et rendez le exécutable, puis exécutez-le.

---

```
#!/bin/sh

SERVER_NAME=`hostname -s`

###
echo creating /etc/rc.d/rc.ws
#this basicly just echos the entire script ;)
echo "#root on nfs stuff

SERVER=$SERVER_NAME

# on a besoin de proc pour mtab, route, etc.
mount -t proc /proc /proc

IP=\`ifconfig eth0|grep inet|cut --field 2 -d ':'|cut --field 1 -d ' '\`

# si le premier montage echoue, c'est qu'on est probablement
# sur le serveur, ou bien que quelque chose ne va pas.
# donc on ne fait la suite que si le premier montage est reussi
mount \$SERVER:/tftpboot/\$IP/sysconfig /etc/sysconfig -o nolock &&
{
   # autres montages
   mount \$SERVER:/home /home -o nolock
   mount \$SERVER:/ /\$SERVER -o ro,nolock

   # creation de /var
   echo Creating /var ...
   mke2fs -q -i 1024 /dev/ram1 1024
   mount /dev/ram1 /var -o defaults,rw
   cp -a /tftpboot/var /

   # configuration reseau
   . /etc/sysconfig/network
   HOSTNAME=\`cat /etc/hosts|grep \$IP|cut --field 2\`
```

```
   route add default gw \$GATEWAY
   ifup lo
}

# restauration des périphériques installés
/etc/rc.d/rc.devfs restore /etc/sysconfig

umount /proc" > /etc/rc.d/rc.ws

###
echo splitting runlevel 3 for the client and server
mv /etc/rc.d/rc3.d /etc/rc.d/rc3.server
cp -a /etc/rc.d/rc3.server /etc/rc.d/rc3.ws
rm /etc/rc.d/rc3.ws/*network
rm /etc/rc.d/rc3.ws/*nfs
rm /etc/rc.d/rc3.ws/*nfsfs
rm /etc/rc.d/rc3.ws/S99local
ln -s /etc/sysconfig/rc.local /etc/rc.d/rc3.ws/S99local
ln -s /etc/rc.d/rc3.server /etc/sysconfig/rc3.d
ln -s /etc/sysconfig/rc3.d /etc/rc.d/rc3.d

###
echo making tmp a link to /var/tmp
rm -fR /tmp
ln -s var/tmp /tmp

###
echo moving various files around and create symlinks for them
echo mtab
/etc/rc.d/init.d/syslog stop
umount /proc
touch /proc/mounts
mount /proc
/etc/rc.d/init.d/syslog start
rm /etc/mtab
ln -s /proc/mounts /etc/mtab
echo fstab
mv /etc/fstab /etc/sysconfig
ln -s sysconfig/fstab /etc/fstab
echo X-config files
mkdir /etc/sysconfig/X11
mv /etc/X11/X /etc/sysconfig/X11
ln -s ../sysconfig/X11/X /etc/X11/X
mv /etc/X11/XF86Config /etc/sysconfig/X11
ln -s ../sysconfig/X11/XF86Config /etc/X11/XF86Config
echo conf.modules
mv /etc/conf.modules /etc/sysconfig
ln -s sysconfig/conf.modules /etc/conf.modules
echo isapnp.conf
mv /etc/isapnp.conf /etc/sysconfig
```

```
ln -s sysconfig/isapnp.conf /etc/isapnp.conf

###
echo creating a template dir for the ws directories
echo /tftpboot/template
mkdir /home/tftpboot
ln -s home/tftpboot /tftpboot
mkdir /tftpboot/template
mkdir /$SERVER_NAME
echo root
ln -s / /tftpboot/template/root
echo sysconfig
cp -a /etc/sysconfig /tftpboot/template/sysconfig
rm -fR /tftpboot/template/sysconfig/network-scripts
ln -s /$SERVER_NAME/etc/sysconfig/network-scripts \
 /tftpboot/template/sysconfig/network-scripts
echo NETWORKING=yes > /tftpboot/template/sysconfig/network
echo `grep "GATEWAY=" /etc/sysconfig/network` >> /tftpboot/template/sysconfig/network
echo "/dev/nfsroot / nfs defaults 1 1" > /tftpboot/template/sysconfig/fstab
echo "none /proc proc defaults 0 0" >> /tftpboot/template/sysconfig/fstab
echo "#!/bin/sh" > /tftpboot/template/sysconfig/rc.local
chmod 755 /tftpboot/template/sysconfig/rc.local
rm /tftpboot/template/sysconfig/rc3.d
ln -s /etc/rc.d/rc3.ws /tftpboot/template/sysconfig/rc3.d
rm /tftpboot/template/sysconfig/isapnp.conf
echo var
cp -a /var /tftpboot/var
rm -fR /tftpboot/var/lib
ln -s /$SERVER_NAME/var/lib /tftpboot/var/lib
rm -fR /tftpboot/var/catman
ln -s /$SERVER_NAME/var/catman /tftpboot/var/catman
rm -fR /tftpboot/var/log/httpd
rm -f /tftpboot/var/log/samba/*
for i in `find /tftpboot/var/log -type f`; do cat /dev/null > $i; done
rm `find /tftpboot/var/lock -type f`
rm `find /tftpboot/var/run -type f`
echo /sbin/fsck.nfs
echo "#!/bin/sh
exit 0" > /sbin/fsck.nfs
chmod 755 /sbin/fsck.nfs

echo all done
```

---

#### 3.2.2 Ajustements manuels

1. Le script de configuration des stations doit être exécuté au tout début de rc.sysinit, donc il faut ajouter les lignes suivantes après avoir défini le PATH :

---

```
# pour les stations montant root par NFS
```

```
/etc/rc.d/rc.ws
```

2. Réduisez /etc/rc.d/rc3.ws à un minimum. Il peut être utile de créer un rc.local.ws, à vous de voir. Réseau et nfs sont déja configurés. Voici d'ailleurs la liste de ce qui a déjà été enlevé/mis à jour par le script :
   - réseau
   - système de fichiers NFS
   - NFS
   - rc.local

## 3.3 Export des systèmes de fichiers appropriés et configuration de bootp

### 3.3.1 Export des systèmes de fichiers

Par exemple ici à l'Université, j'ajouterai ceci à /etc/exports :

```
/ *.st.hhs.nl(ro,no_root_squash)
/home *.st.hhs.nl(rw,no_root_squash)
```

Remplacez les noms de domaine par les vôtres et relancez NFS :

/etc/rc.d/init.d/nfs restart

Pour les utilisateurs de knfsd : il n'est pas possible d'avoir plusieurs exports d'une partition avec des permissions différentes. De même, knfsd ne permet pas de changer de partition (par exemple si un client monte /, et /usr est sur une autre partition, le client n'aura pas accès à /usr). Ainsi, si vous utilisez knfsd, il faudra qu'au moins /home soit sur une partition différente ; le script de préparation du serveur a mis /tftpboot sous /home : il ne nécessite pas une partition supplémentaire. Si vous voulez accéder à d'autres partitions depuis vos clients, exportez les séparément et ajoutez les lignes de montage correspondantes dans /etc/rc.d/rc.ws.

### 3.3.2 Configurer bootp

1. Si bootp n'est pas encore installé, c'est le moment de le faire. Il est inclus dans la RedHat.
2. Editez /etc/inetd.conf et supprimez le commentaire sur la ligne commençant par bootp ; si vous utilisez une bootprom, enlevez également le commentaire pour tftp.
3. Redémarrez inetd :

   /etc/rc.d/init.d/inetd restart

# 4 Ajouter des stations

## 4.1 Créer une disquette de démarrage (bootdisk) ou une bootprom

### 4.1.1 Créer un bootdisk

Même si vous avez l'intention d'utiliser une bootprom, il est plus sage de tester d'abord avec un bootdisk. Pour le créer :

```
dd if=/<path-to-zImage>/zImage of=/dev/fd0
```

### 4.1.2 Créer une bootprom

Il y a plusieurs paquets libres disponibles :

1. netboot, c'est le plus complet. Il utilise les pilotes (packet drivers) DOS standards donc presque toutes les cartes sont supportées. Un truc très utile qui était passé sur la mailing list : compresser les packetdrivers, la plupart des pilotes commerciaux étant trop gros pour tenir dans une bootprom. La documentation de netboot est assez complète : on ne la reprendra pas ici. Avec elle, créer une bootprom et démarrer une station devrait aller de soi. La page web de netboot :
2. etherboot, l'autre package libre Il propose quelques améliorations comme le dhcp. Mais il utilise son propre format de drivers donc supporte moins de cartes. Je ne l'ai pas utilisé donc ne peux en dire plus. La page web :

A propos des roms : la plupart des cartes peuvent recevoir des eproms de 28 pins. Celle-ci ont une taille maximale de 64 ko. Pour la plupart des cartes, on aura besoin de 32 ko avec netboot. Quelques drivers tiendront dans une rom de 16 ko mais la différence de prix est minime. Ces eproms sont standards (on y écrit avec un *eprom burner* ordinaire).

## 4.2 Créer un répertoire station

Il suffit de recopier le répertoire qui sert de modèle (template) en tapant :

```
cd /tftpboot ; cp -a template <ip>
```

On peut aussi, bien sûr, recopier le répertoire d'une station ayant la même souris, carte graphique et moniteur. Dans ce cas la configuration réalisée à l'étape 4.5 est inutile.

## 4.3 Ajouter les entrées dans /etc/bootptab et /etc/hosts

Editer /etc/bootptab et ajouter une entrée pour une station de test, par exemple :

```
nfsroot1:hd=/tftpboot:vm=auto:ip=10.0.0.237:\
:ht=ethernet:ha=00201889EE78:\
:bf=bootImage:rp=/tftpboot/10.0.0.237/root
```

---

Remplacer nfsroot1 par le nom d'hôte de la station. Remplacer 10.0.0.237 par son adresse IP et 00201889EE78 par son adresse MAC. Si vous ne connaissez pas cette dernière, démarrez avec la disquette de boot que vous venez de créer et vous la verrez apparaître dans les messages affichés au boot. Bootpd est certainement déjà lancé, mais pour en être sûr, essayons de le redémarrer :

killall -HUP bootpd

Si cela échoue, c'est qu'il ne tournait pas. Dans ce cas inetd le démarrera au moment voulu.

### 4.4 Démarrer la station pour la première fois

Démarrez simplement la station depuis le bootdisk. Vous devriez avoir ainsi une station en mode texte, avec exactement la même configuration que le serveur exceptés l'adresse IP et les services lancés. Même si vous comptez utiliser une bootprom, il est plus sage de tester d'abord avec un bootdisk.

### 4.5 Configuration spécifique à la station

1. Premièrement, lancez mouseconfig pour installer la souris. Pour appliquer les changements, faites un :

   /etc/rc.d/init.d restart

2. Lancez Xconfigurator ; quand Xconfigurator a détecté la carte et que vous pouvez cliquer sur ok, ne le faites pas ! Comme nous avons déplacé le lien du serveur X de /etc/X11/X vers /etc/sysconfig/X11/X, Xconfigurator ne pourra pas créer le bon lien. Ceci étant, pour être sûr que Xconfigurator continue correctement, basculez sur une autre console et créez le lien sous /etc/sysconfig/X11 vers le serveur X conseillé. Maintenant, quittez Xconfigurator et testez le serveur X.

3. Configuration de tout ce qui diffère du serveur ou du template :

   - son : il sera peut-être nécessaire de modifier isapnp.conf et conf.modules, les deux étant déjà des liens vers /etc/sysconfig (modification faite par le script de prépration du serveur).
   - cdrom : lien sous /dev, entrée dans /etc/fstab, etc.
   - rc.local : faites tous les changements nécessaires

4. Sauvegarde des liens et autres changements effectués sous /dev :

   /etc/rc.d/rc.devfs save /etc/sysconfig

5. Voilà, c'est terminé.

## 5 Bonus : démarrer depuis un cdrom

La plupart des opérations ci-dessus sont valables pour démarrer depuis un cdrom. Comme je voulais également documenter cette façon de booter, je le précise ici pour éviter de taper trop de choses une seconde fois.

Pourquoi démarrer depuis un cdrom ? C'est surtout intéressant partout où l'on veut faire tourner une application spécifique comme un kiosque, une base de données de bibliothèque ou un cyber-café, et qu'on n'a pas de réseau ou de serveur pour utiliser root par NFS.

## 5.1 Principe de base

C'est simple : démarrer avec un cdrom en tant que racine. Pour que ce soit possible, nous utiliserons l'extension rockridge pour graver un système de fichiers unix et l'extension eltorito pour rendre le cd amorçable.

### 5.1.1 Les choses ne peuvent être si simples...

Bien sûr cette configuration soulève quelques problèmes. Ils sont à peu près les mêmes que précédemment :

1. Nous avons besoin d'accès en écriture sur : /dev, /var et /tmp.
   - Nous utiliserons les mêmes solutions :
     - pour /dev nous utiliserons Devfs
     - pour /var et /tmp nous utiliserons un ramdisk partagé de 1 Mo. /tmp est remplacé par un lien vers /var/tmp.
     - le remplissage du ramdisk peut être fait aussi bien à partir d'une archive que d'un répertoire template. Nous retiendrons là encore le répertoire template pour la simplicité des modifications.
2. Certaines applications ont besoin d'un accès à /home en écriture.
   - Dans ce cas, on mettra le répertoire de l'utilisateur de ces applications sous /var, et on finira de remplir /var à chaque boot.
3. /etc/mtab doit être accessible en écriture :
   - Créer un lien vers /proc/mounts et créer un fichier vide sous /proc, comme décrit précédemment.

## 5.2 Créer une configuration de test

1. Pour commencer, prenez une des machines que vous allez utiliser et mettez dedans un gros disque et un graveur de cd.
2. Installez la distribution de votre choix et laissez une partition de 650 Mo pour le test. Cette installation servira à créer l'image iso et à graver le cd, aussi il faut installer les outils necessaires. Elle servira également à recommencer en cas de problème.
3. Sur la partition de 650 Mo, installez la distribution de votre choix avec la configuration que vous voudrez avoir sur le cd. Ce sera la configuration de test.
4. Démarrez sur la configuration de test.
5. Compilez le noyau comme décrit dans la section 3.1, en suivant toutes les étapes. Les modifications pour devfs doivent être faites ici aussi. A l'étape 3, ajoutez ce qui suit :
   - isofs compilé dans le noyau
   - devfs compilé
   - support du cdrom compilé
   - tout ce dont vous avez besoin, compilé ou en module
6. Configuration de la partition de test :
   - créer l'utilisateur qui lancera les applications

- mettre son répertoire sous /var
- installer l'application (si nécessaire)
- configurer l'application si nécessaire
- configurer l'utilisateur de telle façon que l'application démarre automatiquement après le login
- configurer linux pour démarrer une session en tant que cet utilisateur
- configurer tout ce qui doit encore être configuré

7. Vérifiez que la configuration démarre correctement sous l'application et que tout fonctionne bien.
8. Redémarrez sur l'installation principale et montez la partition de 650 Mo sur /test.
9. Mettez ce qui suit dans un fichier /test/etc/rc.d/rc.iso (il sera exécuté au début de rc.sysinit pour créer /var) :

```
#/var
echo Creating /var ...
mke2fs -q -i 1024 /dev/ram1 1024
mount /dev/ram1 /var -o defaults,rw
cp -a /lib/var /

#restore devfs settings, needs proc
mount -t proc /proc /proc
/etc/rc.d/rc.devfs restore /etc/sysconfig
umount /proc
```

10. Editez /test/etc/rc.sysinit en commentant les lignes où / est remonté en lecture-écriture et ajoutez les 2 lignes suivantes après l'initialisation de la variable PATH :

```
#to boot from cdrom
. /etc/rc.d/rc.iso
```

11. Copiez ce qui suit dans un script et exécutez-le : cela va créer un répertoire modèle pour /var et des liens pour /tmp et /etc/mtab.

```
#!/bin/sh
echo tmp
rm -fR /test/tmp
ln -s var/tmp /test/tmp

###
echo mtab
touch /test/proc/mounts
rm /test/etc/mtab
ln -s /proc/mounts /test/etc/mtab

###
echo var
mv /test/var/lib /test/lib/var-lib
```

```
mv /test/var /test/lib
mkdir /test/var
ln -s /lib/var-lib /test/lib/var/lib
rm -fR /test/lib/var/catman
rm -fR /test/lib/var/log/httpd
rm -f /test/lib/var/log/samba/*
for i in `find /test/lib/var/log -type f`; do cat /dev/null > $i; done
rm `find /test/lib/var/lock -type f`
rm `find /test/lib/var/run -type f`
```

12. Enlevez la création de /etc/issue* de /test/etc/rc.local (ça planterait à coup sûr).
13. Maintenant, démarrez sur la partition de test : elle sera en lecture seule comme un cdrom. Si quelque chose ne fonctionne pas, redémarrez sur la partition de travail et réparez puis réessayez. On peut aussi remonter / en lecture-écriture, réparer puis redémarrer directement sur la partition de test. Pour remonter / :

    mount -o remount,rw /

## 5.3 Créer le cd

### 5.3.1 Créer une image de démarrage (image de boot)

D'abord, démarrer sur la partition de travail. Pour créer un cd amorçable, nous aurons besoin d'une image d'une disquette de démarrage. Mais copier par dd une *zimage* ne suffit pas parce que, au tout début du chargement de celle-ci, un pseudo lecteur de disquette est créé et le chargeur du système ne s'y retrouve plus dans le cas d'un cd amorçable. Donc nous utiliserons plutôt syslinux.

1. récupérer boot.img sur un cdrom redhat
2. monter boot.img quelque part par loopback en tapant :

    mount boot.img somewhere -o loop -t vfat

3. enlever tout ce qui est dans boot.img sauf :
    - ldlinux.sys
    - syslinux.cfg
4. copier le noyau de la partition de test vers boot.img
5. editer syslinux.cfg pour ajouter ce qui suit, en remplaçant zImage par le nom d'image approprié :

```
default linux

label linux
kernel zImage
append root=/dev/<insert your cdrom device here>
```

6. démonter boot.img :

umount somewhere

7. Si /etc/mtab est un lien vers /proc/mounts, le démontage ne va pas automatiquement libérer /dev/loop0 donc il faut le libérer en tapant :

   losetup -d /dev/loop0

#### 5.3.2 Créer l'image iso

Maintenant que nous avons l'image de boot et une installation qui peut démarrer sur un montage en lecture seule, il est temps de créer une image iso du cd :

1. copier boot.img sur /test
2. aller dans le répertoire ou vous voulez stocker l'image (en prenant garde qu'il y ait assez de place sur la partition)
3. générer l'image :

   mkisofs -R -b boot.img -c boot.catalog -o boot.iso /test

#### 5.3.3 Vérifier l'image iso

1. monter l'image en loopback en tapant :

   mount boot.iso somewhere -o loop -t iso9660

2. vérifier que le contenu est correct
3. démonter boot.iso :

   umount somewhere

4. si /etc/mtab est un lien sur /proc/mounts, libérer /dev/loop0 :

   losetup -d /dev/loop0

#### 5.3.4 Graver le cd

Si cdrecord est installé et configuré :

cdrecord -v speed=<desired writing speed> dev=<path to your writers generic scsi device> boot.iso

### 5.4 Démarrer sur le cd et le tester

Hé bien le titre de ce paragraphe a tout dit ! ;)

# 6 Remerciements


- La HHS (Haagse Hoge School), l'établissement où j'ai développé et testé cette configuration : elle était utilisée dans plusieurs labos. C'est également là que j'ai écris la première version de ce HowTo.
- ISM : une société néerlandaise où j'ai réalisé mon projet de fin d'études. Une partie de ce projet concernait des machines sans disque, j'ai donc dû pousser un peu plus loin le développement de cette configuration et j'ai eu le temps de mettre à jour ce HowTo.
- A tout ceux qui me donneront des conseils utiles une fois que cette version sera sortie ;)


# 7 Commentaires

Commentaires, suggestions et autres sont les bienvenus et peuvent être adressés à Hans de Goede : j.w.r.degoede@et.tudelft.nl