# Guide pratique de l'amorçage sur réseau et des systèmes de fichiers racine exotiques

Version française du *Network-boot-HOWTO*

Brieuc Jeunhomme <bbp@via.ecp.fr>
Logilab S.A.
Adaptation française: Nicolas Jadot

Version : 0.3.fr.1.0

2004-09-09

Historique des versions

| | | |
|---|---|---|
| Version 0.3.fr.1.0 | 2004-09-09 | NJ |
| Première version française | | |
| Version 0.3 | 2002-04-28 | bej |
| Insertion de retours d'expérience, ainsi que des liens vers différents projets | | |
| Version 0.2.2 | 2001-12-08 | dcm |
| Placé sous licence GFDL | | |
| Version 0.2.1 | 2001-05-21 | logilab |
| Bibliographie terminée | | |
| Version 0.2 | 2001-05-19 | bej |
| Plusieurs amélioration et ajout des retours d'expérience de Ken Yap | | |
| Version 0.1.1 | 2001-04-09 | logilab |
| Premier brouillon public | | |
| Version 0.1 | 2000-12-09 | bej |
| Brouillon initial | | |

**Résumé**

Ce document explique comment mettre rapidement en œuvre un serveur sous linux pouvant fournir tout ce qui est nécessaire à un client linux pour démarrer sur un réseau IP. Il inclut des éléments extraits du Diskless-HOWTO, du Diskless-root-NFS-HOWTO, de la documentation du noyau linux, des pages du projet etherboot et de linux terminal server project, ainsi que de l'expérience personnelle de l'auteur, acquise lorsqu'il travaillait pour Logilab. Eventuellement, ce document peut rendre obsolètes le Diskless-HOWTO et le Diskless-root-NFS-HOWTO. Veuillez noter également que vous trouverez d'autres informations utiles dans le guide pratique "De la mise sous tension à la ligne de commande bash" et dans le Thin-Client-HOWTO, ainsi que sur la page écrite par Claus-Justus Heine sur le basculement entre pages NFS.

## Table des matières

# 1. Introduction

## 1.1. De quoi parlons-nous ?

Les noyaux linux récents permettent de démarrer une machine sous linux entièrement depuis le réseau, en chargeant le noyau et le système de fichiers racine depuis un serveur. Pour ce faire, le poste client peut utiliser diverses méthodes de démarrage : eeproms spécialement configurées, cartes réseaux implantant les protocoles RARP, BOOTP ou DHCP, CD-ROM, lanceurs chargés depuis une disquette ou un disque dur local.

## 1.2. Remerciements

Logilab a soutenu la rédaction de ce guide. Reportez-vous à son site [http://www.logilab.org] pour les nouvelles versions de ce document. Je remercie également les administrateurs et les développeurs de ether-

boot, netboot, plume et linux terminal server project, qui ont réellement rendu possible le démarrage sur réseau d'une station linux.

J'adresse des remerciements particuliers à Ken Yap, membre du projet etherboot, dont les commentaires ont beaucoup aidé à la qualité de ce guide

Je remercie également Jerome Warnier, développeur principal du projet plume, Pierre Mondié, Kyle Bateman, Peter T. Breuer, Charles Howes et Thomas Marteau pour leurs commentaires et leurs contributions.

# 1.3. Plaidoyer pour les stations sans disque

## 1.3.1. Acheter est plus rentable que construire

Parfois, acheter une station linux sans disque sera moins coûteux que de la construire. Consultez les sites dont les adresses figurent en annexe, qui sont ceux de sociétés vendant des cartes réseau ou des machines sans disque pour linux. Ces sociétés vendent des millions d'unités de machines produites en grande série, ce qui en réduit le coût unitaire.

## 1.3.2. Avantages des stations sans disque

Les ordinateurs sans disque deviendront de plus en plus populaires d'ici quelques années. Leur succès ira de paire avec l'arrivée sur le marché de cartes réseau haut débit à bas prix. Une carte offrant un débit de 100 Mégabit (taux de transfert de 12,5 Mo par seconde) est aujourd'hui répandue et, dans un ou deux ans, des cartes offrant des débits de 1000 MBit (125 Mo par seconde) deviendront accessibles et s'imposeront comme le standard.

Dans un futur proche, les fabricants d'écrans y incluront le processeur, la carte réseau et la mémoire afin d'en faire des ordinateurs sans disque. Ce qui permettra d'éliminer le boitier et d'économiser de la place. L'écran sera muni des prises pour clavier, souris, réseau et alimentation électrique.

Les bénéfices à attendre d'une machine sans disque sont les suivants :

- Le coût total de possession d'une station sans disque est très bas. Le coû de possession étant la somme des coût d'achat et de maintenance. Le coût de la maintenance est en généralement égal à trois à cinq fois celui de l'achat de l'ordinateur et augmente d'année en année. Dans le cas d'une solution sans disque, le coût de maintenance est éliminé.
- Toutes les sauvegardes sont centralisées sur un unique serveur central.
- Les onduleurs, conditionnements d'air, protections contre la poussière sont inutiles pour ces machines ; seuls les serveurs en ont la nécessité.
- Une meilleure résistance aux virus - certains virus informatiques ne peuvent s'attaquer à des machines dépourvues de disque. Seul le serveur central a besoin d'être protégé contre ces attaques. Ainsi peut-on économiser plusieurs millions d'Euros en évitant l'installation des anti-virus et le nettoyage des disques.
- Les serveurs peuvent disposer de disques performants de grande capacité, l'utilisation de l'espace de stockage peut être optimisée en le partageant entre les utilisateurs. La tolérance de panne peut être mise en œuvre en utilisant des disques RAID sur le serveur.
- Sur certaines installations, il est possible de partager la mémoire vive du serveur entre les utilisateurs de plusieurs machines sans disque. Si un navigateur web est utilisé par plusieurs personnes, une seule copie de ce programme sera chargée en mémoire.
- Le nombre d'administrateurs système nécessaire à la maintenance d'un unique serveur est très réduit.

- Aucune administration du poste client. Les stations sans disque sont sans maintenance et sans soucis.
- La durée de vie d'une station sans disque est plus élevée.
- Les opérations d'installation et de mise à jour du matériel et du logiciel sont inutiles sur une machine sans disque.
- Les coût inhérents aux CD-ROM, disquettes, bandes, modem, onduleurs, imprimantes, etc, sont éliminés.
- Une machine sans disque peut être utilisée dans des endroits comme des usines, où un disque dur serait trop fragile.

# 1.4. Exigences

## 1.4.1. Exigences matérielles

La chose la plus importante dont il faut disposer pour pouvoir amorçer un ordinateur par le réseau est d'avoir un équipement permettant à la station d'exécuter un gestionnaire d'amorçage , qui chargera le noyau sur le serveur et lançera son exécution. Une autre solution est d'utiliser un noyau local, sur un périphérique quelconque, qui montera un système de fichier racine situé sur le serveur. Il existe de multiples solutions : inscrire les instructions de démarrage sur une EPROM, utiliser une carte réseau supportant BOOTP/DHCP, utiliser une disquette, un disque dur de petite taille ou un CD-ROM pour charger le noyau. Notez que quelques fabricants vendent des ordinateurs en mesure de démarrer sur le réseau : certaines stations Sun implantent le protocole BOOTP.

Les autres caractéristiques matérielles dépendent de l'usage que vous envisagez : sur certains sites, toutes les application s'exécutent du côté serveur, qui doit alors être performant, mais les stations peuvent être très légères : selon l'utilisation, un 80486 dôté de 16 Mo de mémoire vive peut suffire. A l'opposé, si les applications s'exécutent sur le client, les exigences matérielles dépendent totalement de la nature de ces applications. Dans ce cas, un serveur léger peut suffire. Un 80486 avec 32 Mo de mémoire sera suffisant pour un nombre réduit de postes clients, mais davantage de mémoire sera nécessaire pour des installations plus vastes, comptant plusieurs centaines ou milliers de stations. Notez que le processeur du serveur n'est pas crucial pour de telles installations.

## 1.4.2. Exigences logicielles

Les sources de la version 2.0 ou plus récente du noyau Linux sont requises. Les outils indispensables à la compilation du noyau sont également nécessaires (reportez-vous à la documentation du noyau Linux pour plus d'information à ce sujet).

Le démon BOOTP (DHCP serait bien, mais je n'expliquerai pas comment le configurer), un serveur NFS (si vous désirez monter le système de fichiers racine par le réseau), sont requis. Nous aurons besoin également du démon TFTP pour pouvoir charger le noyau à distance. Enfin, l'utilitaire mknbi, fourni avec la distribution etherboot [http://etherboot.sourceforge.net] et, si vous utilisez une EPROM LanWorks, comme celles des cartes 3Com 3C905, vous aurez besoin du programme imggen, disponible à http://www.ltsp.org/contrib/.

# 1.5. Références et documentation associée

Cette documentation a été écrite à l'intention d'administrateurs systèmes expérimentés, rompus à la pratique de linux, comme l'utilisation de grep, sed et awk, la programmation shell, les processus et les scripts de démarrage, la compilation du noyau et la configuration d'un serveur NFS. Avoir l'expérience du passage d'arguments au noyau aidera. Des informations sur ces sujets peuvent être trouvées dans les pages de manuel de grep, sed, awk, dans les pages de manuel et d'info du shell bash, dans les guides pratiques "Disquettes d'amorçage", "De la mise sous tension à la ligne de commande bash", "Noyau linux", "Boot-Param-HOWTO" , dans les pages de manuel de bootparam et de rdev, dans le guide pratique de NFS et sur la page de manuel d'exports.

Il existe de multiples sources d'information concernant le démarrage sur réseau mais, et c'est pourquoi j'ai écrit ce guide, aucune ne décrit toutes les possibilités existantes, et la plupart sont spécifiques à une méthode. Celle qui m'a été la plus utile est la documentation fournie par linux terminal server project [http://www.ltsp.org], bien que je n'ai pas utilisé le paquetage qu'elle recommande et que j'ai choisi de décrire ici comment procéder sans ces paquetages, car ils font le choix d'exécuter toutes les applications à distance sur le serveur. Des informations utiles sont également disponible sur la page du projet etherboot [http://etherboot.sourceforge.net].

Enfin, vous pouvez trouver des informations utiles mais succintes dans l'arborescence des sources du noyau, dans le répertoire `/usr/src/linux/Documentation`, `/usr/src/linux` étant le répertoire où résident les sources du noyau.

## 1.6. Retour d'expérience

J'apprécierai particulièrement les retours d'expérience concernant ce document. N'hésitez pas à m'écrire à `<bbp@via.ecp.fr>` pour me faire part de vos comentaires, corrections ou suggestions. Vous pouvez également utiliser l'adresse `<contact@logilab.fr>`.

## 1.7. Mentions légales



# 2. Panorama du démarrage sans disque

Vous pensez qu'il est temps de passer aux choses sérieuses ? C'est parti.

## 2.1. Récupérer les paramètres IP

On peut se demander comment une station peut démarrer sur un réseau IP sans connaître son adresse IP. En pratique, trois protocoles permettent au client d'obtenir cette information, ainsi que certains paramètres supplémentaires de configuration :

- RARP : le plus simple de ces protocoles. Cependant, je pense qu'il ne permet pas au serveur de spécifier au client comment charger le noyau aussi ne l'utiliserons-nous pas. Dans les faits, il existe une convention utilisant l'adresse IP de la station comme nom de fichier ; un client récupérant l'adresse 192.168.42.12 par RARP téléchargerait le fichier `/tftpboot/192.168.42.12` par tftp, comme le ferait le noyau linux. Le nom du fichier pourrait aussi être la forme hexadécimale de l'adresse IP, mais cette solution est très dépendante de l'implémentation.
- BOOTP : ce protocole autorise un serveur à fournir davantage d'informations à un client (identifié par son adresse MAC), en particulier son adresse IP, le masque de sous-réseau, l'adresse de diffusion, l'adresse réseau, celle de la passerelle, le nom d'hôte et le chemin d'accès au noyau. C'est la solution que nous utiliserons.
- DHCP : une extension de BOOTP.

## 2.2. Charger le noyau

Une fois que le client à récupéré son paramétrage IP, si le noyau n'est pas disponible sur un support local quelconque (disquette, CD-ROM, disque dur), le client le chargera par le protocole TFTP. Son emplacement est fourni par le server BOOTP/DHCP. Un serveur (pas nécessairement le serveur BOOTP/DHCP) devra exécuter le démon TFTP pour les noyaux non locaux. Le noyau obtenu après compilation ne peut pas être utilisé en l'état avec BOOTP/DHCP ; l'image du noyau doit être modifiée grâce à l'utilitaire **mknbi** (et modifiée de nouveau avec l'utilitaire **imggen** si vous utilisez une EPROM LanWorks). L'utilitaire **mknbi** devra également être utilisé pour modifier les noyaux qui seront inscrits sur une ROM.

## 2.3. Monter le système de fichiers racine

Une fois le noyau lancé, il va tenter de monter son système de fichiers racine. Sa localisation est obtenue par BOOTP/DHCP et il est monté grâce à NFS. Cela signifie qu'un client utilisera deux fois BOOTP pour démarrer : la première fois pour charger son noyau, la seconde pour connaître l'emplacement de la racine (qui peut se trouver sur un troisième serveur).

Une autre solution consiste à utiliser un disque virtuel comme racine. Dans ce cas, l'image du disque virtuel est chargée par TFTP.

## 2.4. Terminer la procédure de démarrage

Une fois le système de fichiers racine monté, vous pouvez commencer à respirer : vous pouvez au moins utiliser les lames sh, sed et awk de votre couteau suisse. Vous allez devoir modifier les scripts d'initialisation du système de fichier de vos clients : par exemple, vous devrez retirer tous les disques durs, disquettes ou CD-ROM qui apparaissent dans le fichier `/etc/fstab` (pour vos stations qui en sont dépourvues), il vous faudra inhiber l'activation des partitions de swap (notez qu'il existe une méthode permettant de monter une partition swap par NFS ou sur un périphérique réseau). Si plusieurs clients utilisent la même racine distante, vous devrez aussi générer automagiquement tous les fichiers de configuration réseau au démarrage.

# 3. Construire le noyau

En premier lieu, construire le noyau pour les clients. Je vous suggère d'effectuer cette opération sur le serveur, ce qui sera bien pratique pour l'installation des modules. Utilisez zImage pour réduire la taille du noyau. Incluez tout ce qui est nécessaire mais essayez d'utiliser autant de modules que possible car de nombreuses implantations clientes de BOOTP sont incapables de charger des noyaux volumineux (au moins sur les architectures Intel x86). Incluez le support d'iramdisk, du protocole NFS, de la racine sur NFS, le support de votre adaptateur réseau et celui de l'autoconfiguration IP par BOOTP ; *n'utilisez pas de module pour ces fonctionnalités !* Enfin, si vous avez l'intention d'utiliser la même racine pour plusieurs clients, ajoutez le support du système de fichier ext2 ou autre, ainsi que le support des disques virtuels (un disque virtuel de 16 Mo conviendra sur la plupart des systèmes). Vous pouvez modifier les arguments du noyau comme à l'habitude (voir le guide pratique "BootPrompt-HOWTO" à ce sujet), mais vous pourrez toujours le faire plus tard.

Là, si vous envisagez de faire usage de BOOTP, copiez le noyau zImage sur le serveur. Nous supposerons qu'il réside dans le répertoire `/tftpboot`, que son nom est zImage, que le nom de l'image créée pour BOOTP à partir de zImage est kernel, et que la racine NFS est sous `/nfsroot`.

Saisissez les commandes suivantes sur le serveur (le paquetage mknbi doit être installé) :

```
# cd /tftpboot
```

```
# chmod 0555 zImage
# chown root:root zImage
# mknbi-linux zImage --output=kernel
--rootdir=/nfsroot
```

Si vous utilisez des EPROM LanWorks, saisissez également les commandes suivantes (l'utilitaire imggen est nécessaire) :

```
# mv -f kernel tmpkernel
# imggen -a tmpkernle kernel
# rm -f tmpkernel
```

Le noyau est près à être utilisé avec BOOTP, DHCP ou sur une ROM. Ces opérations ne sont bien entendu pas nécessaires si vous comptez vous servir d'un support local au client.

## 3.1. Racine sur un disque virtuel

Il est possible d'utiliser un disque virtuel pour le système de fichiers racine. Dans ce cas, la commande permettant de modifier l'image binaire du noyau quelque peu différente. Si vous choisissez cette option, il vous faudra permettre le support du disque virtuel initial (initrd), et vous pourrez peut-être vous passer du support de NFS, ou le compiler en tant que module.

Il est temps de s'intéresser à ce qui se passe quand vous utilisez initrd. La documentation complète est disponible dans les sources du noyau dans le fichier `Documentation/initrd.txt`. Je dois vous prévenir que je n'ai jamais essayé :).

Quand initrd est activé, le gestionnaire de démarrage charge en premier le noyau et le disque virtuel initial en mémoire. Le disque virtuel est monté en lecture-écriture comme système de fichiers racine. Le noyau recherche un fichier nommé `/linuxrc` (un exécutable binaire ou un script commençant par #!). Quand `/linuxrc` se termine, la racine traditionnelle est montée sous `/`, et la séquence de démarrage normale est exécutée. Donc, pour permettre à la station de travailler entièrement sur le disque virtuel, il vous suffit de créer un lien de `/linuxrc` vers `/sbin/init`, ou d'écrire là un script exécutant toutes les actions que vous désirez, et d'arrêter l'ordinateur.

Une fois le noyau compilé, vous devez construire un système de fichiers racine pour votre installation. La méthode est expliquée dans la section "Configuration des clients, construction du système de fichiers racine". Je supposerai que cela est déjà fait et que la racine des clients est, temporairement, dans le répertoire `/tmp/rootfs`. Il faut maintenant créer l'image du disque virtuel. Une méthode simple est :

- Assurez-vous que l'ordinateur que vous utilisez dispose du support des disques virtuels et d'un tel périphérique (`/dev/ram0`).
- Créez un système de fichiers vide de la taille appropriée sur ce périphérique :

  ```
  # mke2fs -m0 /dev/ram0 300
  ```

- Montez ce périphérique quelque part :

  ```
  # mount -t ext2 /dev/ram0 /mnt
  ```

- Copiez tout ce qui est nécessaire dans votre nouvelle racine, et créez le futur `/linuxrc` si vous ne l'avez pas fait dans `/tmp/rootfs/linuxrc` :

```
# cp -a /tmp/rootfs/* /mnt
```

- Démontez le disque virtuel :

```
# umount /mnt
```

- Sauvegardez l'image du disque virtuel dans un fichier et libérez la :

```
# dd if=/dev/ram0 of=initrd bs=1024 count=300
# freeramdisk /dev/ram0
```

Ce qui a été noté plus haut concernant les EPROM LanWorks est également valable si vous utilisez initrd.

Maintenant, vous devez modifier l'image du noyau, comme expliqué plus haut, avec l'utilitaire **mknbi-linux**. La suite de commandes diffère peu (je suppose que l'image du noyau est `/tftpboot/zImage` et que l'image du disque virtuel est `/tmp/initrd`) :

```
# cd /tftpboot
# chmod 0555 zImage
# chown root:root zImage
# rdev zImage /dev/ram0
# mknbi-linux zImage --output=kernel
--rootdir=/dev/ram0 /tmp/initrd
```

# 4. Configuration des démons

## 4.1. Démon NFS

Contentez-vous d'exporter le répertoire dans lequel réside le système de fichiers racine du client (référez-vous à la page de manuel d'export pour plus d'information sur cet aspect). Le plus simple est d'exporter avec les attributs no_root_squash et rw, mais une configuration idéale exporterait la plus grande part de la racine avec les attributs root_squash et ro, et listerait, dans le fichier `/etc/exports`, tous les répertoires qui nécessitent réellement les attributs no_root_squash et/ou rw. Commencez avec les attributs rw et no_root_squash pour tout, le réglage fin sera fait plus tard.

Bien sûr, le serveur NFS est inutile si vous envisagez d'utiliser un disque virtuel sur vos clients.

## 4.2. Démon BOOTP

Je supposerai que le paquetage bootpd est installé. Le fichier de configuration par défaut est `/etc/bootptab`, et sa syntaxe est décrite dans la page de manuel de bootptab. Créons ce fichier.

Lancez, en tant que root, votre éditeur favori. Vim. Si, si, c'est celui-là. Si ce n'est pas le cas, il doit le devenir. Maintenant, tapez les lignes suivantes (ce sont les attributs par défaut). Tous les attributs que vous définissez ici, et qui ne sont redéfinis dans aucune liste d'attributs spécifique à une machine, seront valables pour tous les clients :

```
        .default\
                :sm=votre masque de sous-réseau\
                :ds=adresse IP du serveur de noms \
                :ht=ethernet\
                :dn=nom de domaine\
                :gw=adresse IP de la passerelle\
                :sa=adresse IP du serveur TFTP\
                :bf=chemin d'accès à l'image du noyau\
                :rp=chemin d'accès à la racine\
                :hn
```

Bien entendu, tous ces paramètres ne sont pas indispensables, et dépendent de la configuration du réseau et de l'implémentation de BOOTP utilisée, mais cela fonctionnera dans la plupart des cas.

Maintenant, ajoutez une entrée pour chacun des clients de votre réseau. Une entrée se présente sous la forme :

```
        nom du client\
                :ha=adresse MAC du client\
                :ip=adresse IP du client
```

L'adresse MAC est l'adresse matérielle du client en hexadécimal, sans les ':'.

Voici un exemple de fichier `/etc/bootptab` :

```
        .default\
                :sm=255.255.0.0\
                :ds=192.168.0.2\
                :ht=ethernet\
                :dn=frtest.org\
                :gw=192.168.0.1\
                :sa=192.168.0.2\
                :bf=/tftpboot/kernel\
                :rp=/nfsroot\
                :hn

        foo\
                :ha=001122334455\
                :ip=192.168.2.12

        bar\
                :ha=00FFEEDDCCBB\
                :ip=192.168.12.42\
                :ds=192.168.2.42
```

Enfin, lancez le démon bootpd avec la commande bootpd -s (il est judicieux d'ajouter cette commande dans vos scripts de démarrage), ou ajoutez la ligne suivante à votre fichier `/etc/inetd.conf` :

```
bootps dgram udp wait root /usr/sbin/tcpd bootpd -i -t 120
```

Si vous désirez tester le serveur BOOTP, ajoutez une entrée dans le fichier `/etc/bootptab` et utilisez le programme bootptest.

## 4.3. TFTP

Configurer le serveur TFTP n'est pas la partie difficile : installez le paquetage tftpd si vous l'avez, et ajoutez la ligne suivante à votre fichier `/etc/inetd.conf` (de nouveau, je suppose que `/tftpboot` est le répertoire contenant l'image du noyau) :

```
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd /tftpboot
```

N'oubliez pas de modifier les droits sur le répertoire `/tftpboot` (chmod 555) le serveur TFTP n'enverra pas les fichiers s'ils ne sont pas lisible par tout le monde.

Vous devez être conscient des limitations imposées par l'utilisation du démon TFTP à partir d'inetd. La plupart des implémentations arrêterons un service s'il est appelé trop souvent . Aussi, si vos clients sont nombreux, cherchez plutôt du côté d'xinetd, ou lancez le démon TFTP isolément.

Maintenant que tous les démons sont correctement configurés, vous pouvez relancer inetd et boire un café. N'oubliez pas de dire à tout le monde que la configuration du serveur est finie, pour pouvoir passer pour un héros avant d'entamer la construction du système de fichiers racine de vos clients.

# 5. Configuration des clients, construction du système de fichiers racine

Fatigué ? Non, vous ne l'êtes pas. Souvenez-vous que vous êtes un héros. Ici commence la partie délicate. Nous allons (err... *vous* allez) construire la racine des clients. Ce ne devrait pas êtres très difficile, mais préparez vous à de nombreux essais et erreurs.

La méthode la plus simple pour créer le système de fichiers racine est d'utiliser un système de fichiers existant et de l'adapter aux besoins. Bien sûr, vous pouvez le construire à la main (comme au bon vieux temps) si cela vous dit :=), mais je n'expliquerai pas cela ici.

## 5.1. Créer les premiers fichiers et répertoires

Tout d'abord, déplacez vous dans le futur répertoire racine de vos stations. Vous pouvez facilement créer le répertoire `/home` par la commande **mkdir**, ou en le copiant à partir d'où vous voulez (utilisez la commande **cp -a** pour effectuer une copie récursive préservant les propriétaires, groupes, liens symboliques et permissions). Même chose pour les répertoires `/mnt`, `/root`, `/tmp` (n'oubliez pas **chmod 0** sur ce dernier, puisqu'il n'est qu'un point de montage du `/tmp` réel que nous utiliserons, car chaque station a besoin de son propre répertoire `/tmp`). Copiez les répertoires `/bin`, `/sbin`, `/boot` et `/usr` existant dans le futur répertoire racine (commande **cp -a**). Vous pouvez créer le répertoire `/proc` avec mkdir, et utiliser **chmod 0** dessus. Notez que certaines applications ont besoin d'un accès en écriture sur le répertoire de leur utilisateur.

Le répertoire `/lib` peut être copié sans danger de n'importe où, mais vous devrez y installer les modules qui conviennent. Pour ce faire, utilisez les commandes suivantes (considérant que le noyau compilé pour les clients réside sur le serveur dans `/usr/src/linux`, et que la racine est dans `/nfsroot`) :

```
# cd /usr/src/linux
# make modules_install INSTALL_MOD_PATH=/nfsroot
```

N'oubliez pas de placer le fichier `System.map` dans le répertoire `/nfsroot/boot`. Un premier problème que nous aurons à résoudre est que, en fonction de votre configuration, le système peut essayer de lancer fsck sur la racine au démarrage. Impossible sans disque dur. La plupart des distributions omettrons fsck si un fichier `fastboot` est présent dans le répertoire racine. Aussi, tapez les commandes suivantes si vous ne comptez pas monter de disque dur :

```
# cd /nfsroot
# touch fastboot
# chmod 0 fastboot
```

Une autre méthode consiste à indiquer à **fsck** que la vérification d'un système de fichiers NFS réussit à chaque fois :

```
# cd /nfsroot/sbin
# ln -s ../bin/true fsck.nfs
```

Le répertoire `/dev` peut être copié d'un autre emplacement vers `/nfsroot`. Les permissions et les liens symboliques devant être préservés, utilisez la commande **cp -a**. Une autre solution est d'utiliser la fonctionnalité devfs des noyaux 2.2.x, qui réduit la consommation mémoire tout en augmentant les performances, mais le défaut de cette méthode est que tous les liens symboliques existant dans `/dev` seront perdus. Le point à garder à l'esprit est que chaque station a besoin de son propre répertoire `/dev`, aussi vous devrez le copier sur un disque virtuel si vous voulez avoir plusieurs clients et ne pas utiliser devfs.

## 5.2. Les répertoires `/var` et `/etc`

Nous utiliserons des disques virtuels pour ces répertoires, chaque client devant avoir les siens propres. Mais nous en avons encore besoin au début pour créer leur structure standard. Notez que cela n'est pas nécessaire si vous n'avez qu'un seul client. Copiez donc ces répertoires (**cp -a**) dans `/nfsroot`. Vous pouvez alors faire un peu de ménage dans `/var` : vous pouvez effacer tout le contenu de `/nfsroot/var/log` et de `/nfsroot/var/run`. Vous pouvez probablement effacer également le contenu de `/nfsroot/var/spool/mail` si vous comptez exporter ce répertoire via NFS. Vous devrez enfin supprimer les fichiers contenant des informations spécifiques à la machine présents dans `/nfsroot/etc` pour les construire à la volée durant le processus de démarrage.

Les scripts de démarrage doivent être modifiés afin de pouvoir monter certaines parties de l'arborescence : le répertoire `/dev` si vous n'utilisez pas devfs, les répertoires `/tmp`, `/var` et `/etc`. Le code suivant effectue ces modifications :

```
# cette partie uniquement si vous n'utilisez pas devfs
mke2fs -q -i 1024 /dev/ram0 16384
mount -n -t ext2 -o rw,suid,dev,exec, \
    async,nocheck /dev/ram0 /dev
# cette partie dans tous les cas
mke2fs -q -i 1024 /dev/ram1 16384
mount -n -t ext2 -o rw,suid,dev,exec, \
    async,nocheck /dev/ram1 /tmp
chmod 1777 /tmp
```

```
cp -a /etc /tmp
mke2fs -q -i 1024 /dev/ram2 16384
mount -n -t ext2 -o rw,suid,dev,exec, \
    async,nocheck /dev/ram2 /etc
find /tmp/etc -maxdepth 1 -exec cp -a '{}' /etc ';'
mount -f -t ext2 -o rw,suid,dev,exec, \
    async,nocheck,remount /dev/ram2 /etc
mount -f -o remount /
cp -a /var /tmp
mke2fs -q -i 1024 /dev/ram3 16384
mount -t ext2 -o rw,suid,dev,exec, \
    async,nocheck /dev/ram3 /var
find /tmp/var -maxdepth 1 -exec cp -a '{}' /var ';'
```

Dans le cas où vous aurez plus d'un seul client, vous devrez changer dynamiquement au démarrage certains fichiers contenus dans `/etc` : ceux contenant l'adresse IP et le nom d'hôte du client. Ces fichiers dépendent des distributions mais vous les retrouverez aisément avec grep. Contentez-vous de retirer de ces fichiers toutes les informations spécifiques à un client, et ajouter le code permettant la génération de ces informations lors du démarrage, *mais uniquement une fois que le nouveau `/etc` a été monté sur le disque virtuel !* Ci-dessous une méthode permettant d'obtenir l'adresse IP et le nom d'hôte lors du démarrage (si bootpc est installé dans l'arborescence des stations) :

```
IPADDR="$(bootpc | awk '/IPADDR/ \
                        {
                          match($0,"[A-Za-z]+")
                          s=substr($0,RSTART+RLENGTH)
                          match(s,"[0-9.]+")
                          print substr(s,RSTART,RLENGTH)
                        }
                       ')"

HOST="$(bootpc | awk '/HOSTNAME/ \
                      {
                        match($0,"[A-Za-z]+")
                        s=substr($0,RSTART+RLENGTH)
                        match(s,"[A-Za-z0-9-]+")
                        print substr(s,RSTART,RLENGTH)
                      }')"

DOMAIN="$(bootpc | awk '/DOMAIN/ \
                        {
                          match($0,"[A-Za-z]+")
                          s=substr($0,RSTART+RLENGTH)
                          match(s,"[A-Za-z0-9-.]+")
                          print substr(s,RSTART,RLENGTH)
                        }')"
```

C'est une solution compliquée mais je pense qu'elle fonctionne sur la plupart des sites. On peut trouver l'adresse IP grâce à la commande **ifconfig** et le nom d'hôte par la commande **host** mais cela n'est pas portable, car les sorties diffèrent entre les systèmes, en fonction de la distribution utilisée et du paramétrage local.

Là, le nom d'hôte pourra être déterminé avec la commande **hostname $HOSTNAME**. Une fois ceci fait, il est temps de générer à la volée les fichiers de configuration contenant l'adresse IP ou le nom d'hôte du client.

## 5.3. Derniers détails

On peut maintenant passer aux réglages fins. Comme `/var` sera monté sur un disque virtuel (suaf si vous n'avez qu'un seul client), vous devrez sauvegarder les logs sur un serveur, si vous voulez les conserver. Une méthode pour ce faire consiste à supprimer le fichier `/nfsroot/etc/syslog.conf` et à le remplacer par le fichier suivant (man syslog.conf pour plus de détails) :

```
*.*     /dev/tty12
*.*     nom ou adresse IP du serveur contenant les logs
```

Avec cette méthode, le serveur de logs devra faire tourner **syslogd** avec l'option -r (reportez-vous à la page de manuel de syslogd).

Si vous utilisez **logrotate** et avez effectué les opérations précédentes, vous devrez remplacer le fichier de configuration de **logrotate** (`/etc/logrotate.conf` sur la plupart des machines) par un fichier vide :

```
# rm -f /etc/logrotate.conf
# touch /etc/logrotate.conf
```

Si vous n'en faites pas usage, contentez-vous de retirer les scripts de rotation des logs de la crontab et, tant que vous n'aurez pas de fichiers de log dans `/var/log`, placez **exit 0** au début de vos scripts de rotation.

Retirez du fichier `/nfsroot/etc/fstab` toutes les références à des disques durs, lecteurs de disquettes ou CD-ROM qui sont absents de vos stations clientes. Ajoutez une entrée pour le répertoire `/var/spool/mail`, qui pourrait être exporté par le serveur par NFS ou par tout autre système de fichiers réseau. Vous pourrez vouloir également ajouter une entrée pour `/home` dans ce fichier.

Vous pouvez également mettre en commentaire les lignes exécutant newaliases, activant le swap, et exécutant depmod -a et également supprimer le fichier `/nfsroot/etc/mtab`. Retirez les commentaires des lignes de vos fichiers de démarrage concernant `/fastboot`, `/fsckoptions`, et `/forecfsck`. Retirez également ou mettez en commentaires toutes les lignes des scripts de démarrage qui tenteraient d'écrire sur le système de fichiers racine, à l'exception des accès en écriture réellement nécessaires, qui devraient tous être redirigés vers l'emplacement d'un ramdisk si vous utilisez plusieurs clients.

## 5.4. Essais...

Le temps est venu d'un petit essai. SAUVEGARDEZ TOUS LES `/nfsroot` NOUVELLEMENT CREES. tar -cvvIf devrait faire cela à merveille. Prenez également le temps de vérifier que vous n'avez rien oublié. Et tentez de démarrer un client.

## 5.5. Une erreur !

Regardez attentivement l'écran de la station cliente pendant la phase de démarrage. Excusez-moi ! j'ai omis de vous dire de connecter un écran... Cours, Forest ! Cours et rapportes-en un. Des messages d'erreur vont probablement s'afficher. Résolvez les problèmes et effectuez de fréquentes sauvegardes du répertoire /

`nfsroot`. Un jour, le client démarrera proprement. Pour l'heure, vous allez devoir résoudre les problèmes apparaissant pendant la séquence d'arrêt ;=P.

# 6. Différentes manières de récupérer le noyau

Nous avons parlé plus haut de la configuration du client et du serveur en vue des opérations s'exécutant une fois que le client a terminé la requête bootp, mais le premier soucis est que la plupart des ordinateur est incapable de fonctionner par défaut comme client bootp. Nous allons voir dans cette section comment résoudre ce problème.

## 6.1. Interfaces réseaux à capacités bootp ou DHCP

C'est le cas le plus simple : certaines cartes réseau fournissent une extension du BIOS contenant un client bootp ou DHCP, aussi il suffit de configurer les opérations bootp ou DHCP dans le BIOS, et tout est dit.

## 6.2. Noyau sur disquette ou disque dur local

Ces deux cas de figure sont assez simples : le noyau est chargé depuis le support local, et tout ce qu'il doit faire est de récupérer ses paramètres réseau par bootp et de monter le système de fichiers racine par NFS ; cela ne devrait poser aucun problème. D'autre part, un disque dur local est un excellent endroit pour loger les répertoire `/var`, `/tmp`, et `/dev`...

Si vous disposez d'un disque dur local, tout ce que vous avez à faire est d'utiliser lilo ou votre chargeur de démarrage préféré, comme d'habitude. Si vous utilisez une disquette, vous pouvez utilisez le chargeur, ou simplement y écrire le noyau : un noyau est directement chargeable. Vous pouvez utiliser une commande comme :

```
# dd if=zImage of=/dev/fd0 bs=8192
```

Cependant, Alan Cox a écrit dans une discussion au sujet du noyau linux que cette fonctionnalité sera retirée tôt ou tard, aussi devrait vous un jour utiliser un chargeur de démarrage également sur les disquettes. Je sais que cela fonctionne toujours avec un noyau 2.4.11, mais le support semble en avoir été retiré dans la version 2.4.13. Voyez le chapitre six du boot-disk-HOWTO [http://www.traduc.org/docs/HOWTO/vf/Bootdisk-HOWTO.html] à ce sujet.

## 6.3. Chargeur de démarrage sans noyau sur un support local

Certains chargeurs de démarrage ont des capacités réseau, aussi devriez vous utilisez ceux là pour chargeur l'image du noyau par le réseau. Quelques-un sont listés ci-dessous :

- netboot [http://netboot.sourceforge.net], chargeur dédié au démarrage réseau.
- GRUB [http://www.gnu.org/software/grub/], le chargeur du projet GNU (GRand Unified Bootloader), de portée vaste.

## 6.4. Créer les ROM des clients

De nombreuses cartes réseau comportent un slot permettant l'insertion d'une EPROM contenant du code BIOS additionnel. Cela permet d'ajouter, par exemple, des fonctionnalités bootp au BIOS. Pour ce faire, il convient tout d'abord de trouver comment activer la prise supportant l'EPROM. Il peut être nécessaire d'agir sur un cavalier ou d'utiliser un logiciel spécifique. Quelques cartes, telles la 905B de 3Com, disposent de branchements pour EEPROM, ce qui permet de modifier le logiciel embarqué dans l'EEPROM. En annexe, vous trouverez des informations sur les EPROM et sur les différents types de composant de mémoire.

Pour obtenir la liste des fabricants de brûleurs d'EPROM, reportez-vous au site Yahoo, dans le répertoire Economie->Entreprises->Matériel->Périphériques->Programmateurs [http://dir.yahoo.com/Business_and_Economy/Companies/Computers/Hardware/Peripherals/Device_Programmers/] ou reportez-vous au guide pratique Diskless-HOWTO, section *Liste des fabricants de brûleurs d'EPROM*.

Si vous choisissez de créer vos propres ROM, vous devrez y charger un logiciel bootp ou DHCP et alors, vous vous retrouverez dans la configuration d'une carte réseau possédant ces caractéristiques, tel que décrit plus haut.

Vous devrez également trouver la taille et la vitesse d'EPROM appropriés à votre adaptateur réseau. Des méthodes pour ce faire sont fournies en annexe, car les fabricants de cartes fournissent rarement ces informations.

### 6.4.1. EPROM LanWorks

Cette précision peut vous sauver. Pour permettre à une EPROM LanWorks BootWare (tm) de démarrer correctement une image du noyau linux, la partie "bootsector" de l'image doit être modifiée pour permettre à la PROM de "sauter" directement à l'adresse de démarrage de l'image. L'image créée par l'utilitaire ‘mknbi-linux‘ est différente et ne fonctionnera pas avec une PROM BootWare.

Vous pouvez trouver le bootsector modifié ainsi qu'un Makefile permettant la création d'une image compatible BootWare aux adresses :

- Paquetage Bwimage : ftp://ftp.ipp.mpg.de/pub/ipp/wls/linux/bwimage-0.1.tgz
- Voir aussi http://www.patoche.org/LTT/net/00000096.html
- ROM de démarrage LanWorks BootWare : http://www.3com.com/lanworks

Reportez-vous au fichier README pour les détails concernant l'installation. Le plus souvent, seules les images du type "zImage" sont supportées. Malheureusement, les paramètres noyau sont ignorés.

Cette section a été initialement écrite par Jochen Kmietsch pour le guide pratique "Machines sans disque" ; posez vos questions par courrier électronique à `<jochen.kmietsch@tu-clausthal.de>`.

## 6.5. CD-ROM local

Cette section a originellement été écrite par Hans de Goede (`<j.w.r.degoede@et.tudelft.nl>`) pour le guide pratique "Système de fichiers racine sur NFS". Je l'ai juste modifié dans la mesure nécessaire pour rendre compte des différences entre le présent document et le guide pratique "Système de fichiers racine sur NFS".

La majeure partie de ce qui est écrit ci-dessus est également valable pour le démarrage par CD-ROM. Pourquoi vouloir démarrer une machine depuis le lecteur de CD-ROM ? Le démarrage sur un CD-ROM est surtout intéressant pour des applications très spécifiques, comme un kiosque, une base de données de bibliothèque, un net-café, ou pour celui qui n'a pas de réseau ou de serveur pour permettre l'usage d'un système de fichiers racine sur réseau.

### 6.5.1. Créer une configuration test

Maintenant que nous savons ce que nous voulons faire et comment, il est temps de réaliser une configuration test.

- Au début, premez seulement l'une des machines que vous désirez utiliser et branchez dessus un gros disque dur et un graveur de CD.
- Installez le linux de votre choix sur cette machine, en veillant à conserver une partition libre de 650 Mo. Cette installation servira à la création d'une image ISO et au gravage d'un CD-ROM, aussi prévoyez les logiciels nécessaires. Elle servira également à réparer les défauts qui empêcheraient le démarrage de la configuration test.
- Sur la partition de 650Mo, installez linux avec la configuration que vous désirez obtenir sur le CD ; ce sera la configuration de test.
- Démarrez la configuration test.
- Compilez un noyau avec les supports isofs et cdrom inclus (pas en modules).
- Configurez l'installation de test comme décrit plus haut, avec le système de fichiers racine en lecture seule.
- Vérifiez que la configuration de test démarre automagiquement et que tout fonctionne.
- Démarrez votre installation principale (la première, voir ci-dessus) et montez la partition de 650Mo dans le répertoire `/test` de l'installation principale.
- Insérez les lignes suivantes dans un fichier nommé `/test/etc/rc.d/rc.iso` qui sera exécuté au démarrage de `rc.sysinit` pour créer le répertoire `/var` :

```
#/var
echo Creating /var ...
mke2fs -q -i 1024 /dev/ram1 16384
mount /dev/ram1 /var -o defaults,rw
cp -a /lib/var /
```

- Editez le fichier `/test/etc/rc.sysint`, mettez en commentaires les lignes dans lesquelles le système de fichier racine est remonté en lecture-écriture (rw), et ajoutez les deux lignes suivantes directement après la configuration de PATH :

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

- Copiez les lignes suivantes dans un script et exécutez le afin de créer un répertoire `/var` générique et créer les liens vers `/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`
```

- Retirez ce qui a trait à la création de `/etc/issue*` du fichier `/test/etc/rc.local` : ce ne peut qu'échouer.
- Maintenant, démarrez la partition de test de nouveau, elle sera lue exactement comme si elle était sur CD-ROM. Si quelque chose ne fonctionne pas, redémarrez l'installation de travail pour résoudre le problème et relancez l'installation de test de nouveau. Pour remonter la racine (`/`) en lecture-écriture (rw), tapez :

```
# mount -o remount,rw /
```

## 6.5.2. Créer le CD

S'il vous faut davantage d'informations que celles fournies ci-dessous, référez-vous au guide pratique "Gravure de CD-ROM".

### 6.5.2.1. Créer une image lançable

En premier lieu, démarrez l'installation de travail. Pour créer un CD lançable, nous avons besoin d'une disquette lançable. Utiliser simplement dd sur un fichier zImage ne fonctionne pas car le chargeur situé au début du fichier zImage ne semble pas apprécier le "faux lecteur de disquette" créé par un CD lançable. Aussi allons-nous utiliser syslinux à la place.

- Récupérez le fichier boot.img d'un CD RedHat
- Montez ce fichier n'importe où au travers de l'interface loopback :

```
# mount boot.img nimporteoù -o loop -t vfat
```

- Retirez tout de boot.img, à l'exception de ce qui concerne ldlinux.sys et syslinux.cfg.
- Copiez l'image du noyau de la partition de test dans boot.img.
- Editez le fichier `syslinux.cfg` pour qu'il contienne les lignes suivantes, en remplaçant bien entendu `zImage` par le nom de votre image :

```
default linux

label linux
kernel zImage
append root=/dev/<le nom de périphérique de votre
lecteur de CD-ROM>
```

- Démontez boot.img :

```
# umount nimporteoù
```

- Si `/etc/mtab` est un lien vers `/proc/mounts`, umount ne libérera pas automagiquement `/dev/loop0`, aussi libérez le par :

```
# losetup -d /dev/loop0
```

### 6.5.2.2. Créer l'image ISO

Maintenant que nous avons une image lançable et une installation capable de démarrer sur un support en lecture seule, il est temps de créer l'image ISO du CD :

- Copiez boot.img dans `/test`
- Déplacez vous dans le répertoire où vous désirez créer l'image et assurez vous que la partition en question dispose de suffisament d'espace libre.
- Créez l'image avec :

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

### 6.5.2.3. Vérifier l'image ISO

- Montez l'image par l'intermédiaire du périphérique loopback :

```
# mount boot.iso quelquepart -o loop -t iso9660
```

- Démontez boot.iso :

```
# umount somewhere
```

- Si `/etc/mtab` est un lien vers `/proc/mounts`, umount ne libérera pas automagiquement `/dev/loop0`, aussi libérez le par :

```
# losetup -d /dev/loop0
```

#### 6.5.2.4. Ecrire le véritable CD

Considérant que vous disposez de **cdrecord** installé et correctement configuré pour votre graveur, tapez :

```
# cdrecord -v speed=<vitesse de
gravure> dev=<chemin au périphérique générique scsi
de votre graveur> boot.iso
```

### 6.5.3. Démarrer et tester le CD

Tout est dit dans le titre ;)

# 7. Comment créer des stations MS-Windows sans disque ?

Comme MS-Windows n'apporte pas de support pour le démarrage sans disque, un simple tour d'horizon des solutions est présenté ici : la solution est d'utiliser un logiciel comme VMWare [http://www.vmware.com] ou son alternative libre, plex86 [http://www.plex86.org]. Bien que plex86 semble avoir été abandonné, on peut encore démarrer certaines versions de MS-Windows en utilisant ce logiciel. Cela permet d'exécuter de manière transparente MS-Windows sur une machine linux.

# 8. Problèmes, trucs, difficultés et liens utiles

## 8.1. Traiter les fichiers spécifiques aux stations de manière transparente

Les sections précédentes ont exposé une manière simple de traiter les fichiers et les répertoires comme `/var`, spécifiques aux différentes stations. La plupart sont simplement construits à la volée et stockés sur ramdisk, mais vous pouvez préférer traiter ce problème sur le serveur NFS. Le projet clusternfs fournit un serveur de fichiers réseau capable de délivrer des fichiers différents en fonctions de critères comme l'adresse IP ou le nom d'hôte. L'idée de base est que, si un client d'adresse IP 10.2.12.42 requiert le fichier, disons, `monfichier`, le serveur cherche un fichier nommé `monfichier$$IP=10.2.12.42$$` et le renvoi à la place du fichier `monfichier` s'il est disponible.

## 8.2. Réduire l'occupation de la mémoire des stations

Une manière simple de diminuer la consommation de mémoire est de disposer plusieurs répertoires créés de manière dynamique sur le même disque virtuel. Par exemple, supposons que le premier disque virtuel contient le répertoire `/tmp`. Alors, on peut déplacer le répertoire `/var/tmp` sur ce disque virtuel avec la commande suivante, exécutée depuis le serveur :

```
# mkdir /nfsroot/tmp/var
# chmod 0 /nfsroot/tmp/var
# ln -s /tmp/var /nfsroot/var/tmp
```

Une autre bonne méthode pour réduire la consommation mémoire si vous ne disposez pas de disque dur local et n'utilisez pas de partition d'échange par réseau est de désactiver l'option **Echange sur périphériques en mode bloc** pendant la compilation du noyau.

## 8.3. Partition swap sur NFS

Si vos stations ne dispose pas de suffisament de mémoire et n'ont pas de disques locaux, vous pouvez utiliser une partition d'échange sur NFS. Cependant, gardez à l'esprit que le code de cette fonctionnalité est encore en phase de développement et que son fonctionnement est généralement assez lent. La documentation complète de cette fonctionnalité est disponible à http://www.instmath.rwth-aachen.de/~heine/nfs-swap/.

La première chose à faire pour mettre en œuvre cette fonctionnalité est de modifier votre noyau (vous devez disposer de la version 2.2 ou supérieure). Téléchargez le modificatif à l'adresse fournie ci-dessus et déplacez vous dans le répertoire `/usr/src/linux`. On considère que le modificatif est dans `/usr/src/patch`. Utilisez la commande suivante :

```
        # cat ../patch | patch -p1 -l -s
```

Ceci fait, compilez votre noyau de manière habituelle en activant les options **Partiton d'echange sur réseau (EXPERIMENTALE)** et **Partiton d'échange par NFS (EXPERIMENTALE)**.

Exportez alors un répertoire en lecture-écriture et avec l'option no_root_squash depuis le serveur NFS. Configurez les clients pour qu'ils montent ce répertoire quelque part (disons dans `/mnt/swap`). Il devra être monté avec des paramètres rsize et wsize inférieurs à la taille de page utilisée par le noyau (soit 4ko sur architecure Intel), sans quoi la machine risque de se trouver à cours de mémoire, pour cause de fragmentation ; reportez vous à la page de manuel de nfs pour les détails concernant les paramètres rsize et wsize. Pour créer une partition d'échange de 20Mo, utilisez les lignes de commande suivantes (à placer dans les scripts d'initialisation des stations clientes) :

```
        # dd if=/dev/zero of=/mnt/swap/swapfile bs=1k count=20480
 # mkswap /mnt/swap/swapfile
 # swapon /mnt/swap/swapfile
```

Bien entendu, ceci n'est valable que pour une seule machine, car si vous en disposez de plusieurs il vous faudra modifier les noms des fichiers et des répertoires, sans quoi toutes vos stations utiliseraient le même fichier d'échange...

Un mot encore sur les caractéristiques des partitions d'échange par NFS : en premier lieu le mécanisme est généralement lent, sauf à disposer d'interfaces réseau spécialement rapides. Cette possibilité n'a pas encore été beaucoup testée. Enfin, ce n'est pas sécurisé : n'importe qui sur le réseau est à même de lire les données transférées.

## 8.4. Echange sur un périphérique réseau en mode bloc (NBD - Network Block Device)

Bien que je n'ai jamais testé personnellement, j'ai eu des retours selon lesquels cette astuce fonctionne, du moins avec des noyaux récents.

Le principe général est le même qu'avec NFS. Le point positif est qu'il n'est pas nécessaire de modifier le code du noyau. Mais la plupart de ses caractéristiques s'appliquent également à cette méthode.

Pour créer un espace de swap de 20 Mo, vous devrez d'abord le créer sur le serveur, puis l'exporter vers le client, enfin, exécuter **mkswap** sur le fichier. Notez que la commande **mkswap** doit être exécutée sur le serveur, car mkswap fait usage d'appels système non supportés par NBD. De plus, cette commande doit être exécutée après que le serveur ait exporté le fichier, car les données contenues dans celui-ci risqueraient d'être détruite au moment où le serveur commence l'exportation. En considérant un serveur nommé NBDserveur, un client nommé NBDclient et que le port TCP utilisé pour l'exportation est le port 1024, la commande à exécuter sur le serveur est :

```
        # dd if=/dev/zero of=/swap/swapfile bs=1k count=20480
 # nbd-server NBDclient 1024 /swap/swapfile
 # mkswap /swap/swapfile
```

Maintenant, le client doit utiliser la commande suivante :

```
# swapon /dev/nd0
```

De nouveau, ce ne sont là que les principes généraux. Les noms de fichiers peuvent également être dépendants des noms ou adresses IP des stations.

Une autre solution de fichier d'échange est de créer un système de fichiers ext2 sur le NBD, d'y créer un simple fichier et, enfin, d'utiliser les commandes **mkswap** et **swapon** pour utiliser le fichier comme fichier d'échange. La seconde méthode est plus proche de celle du fichier d'échange sur NFS que la première.

## 8.5. Se débarasser des messages d'erreur au sujet de `/etc/mtab` ou de répertoires non démontés lors de la séquence d'arrêt;

Les commandes suivantes, exécutées sur le serveur, devraient résoudre le problème :

```
# ln -s /proc/mounts /nfsroot/etc/mtab
# touch /nfsroot/proc/mounts
```

## 8.6. Installer de nouveaux paquetages sur les stations

Une méthode simple est d'utiliser, du côté serveur, un chroot et d'exécuter alors votre utilitaire d'installation favori. Pour changer la racine vers l'endroit approprié, utiliser la commande suivante :

```
# chroot /nfsroot
```

Les utilisateurs de distributions Debian seront particulièrement intéressés par l'option --root de dpkg, qui indique à dpkg où est la racine du système cible.

# A. Composants mémoire persistants

Cette section décrit brièvement les différents types de composants mémoire :

- PROM : prononcé prom, acronyme anglo-saxon pour *programmable read-only memory* (mémoire programmable à accès en lecture seule). Une PROM est un composant mémoire sur lequel les données peuvent être écrites une seule fois. Une fois qu'un programme est inscrit sur une PROM, il y demeure. Contrairement à la RAM, les PROM conservent leurs contenus quand l'ordinateur est hors tension. La différence entre une PROM et une ROM (*read-only memory* - mémoire en lecture seule) est que la PROM est vide en sortie de chaîne , alors que la ROM reçoit son contenu lors du processus de fabrication. Pour inscrire des données sur une PROM, il est nécessaire de disposer d'un équipement appelé programmateur, ou brûleur, de PROM. Le processus de programmation d'une PROM est parfois appelé brûler une PROM. Une EPROM (*erasable programmable read-only memory* - mémoire programmable et effaçable à accès en lecture seule) est un type particulier de PROM dont le contenu peut être effacé par exposition à un éclairage ultraviolet. Une fois effacée, elle peut être reprogrammée. Une EEPROM est similaire à une PROM, mais son effacement se fait électriquement.

- EPROM : acronyme anglo-saxon de *erasable programmable read-only memory* (mémoire programmable et effaçable à accès en lecture seule) et prononcé e-prom ; l'EPROM est un type particulier de mémoire non-volatile effaçable par exposition aux ultraviolets. Les ultraviolets effacent le contenu du composant, rendant possible sa reprogrammation. Pour écrire sur ou pour effacer une EPROM, il faut disposer d'un programmateur (ou brûleur) de PROM. L'EPROM diffère de la PROM par le fait qu'une PROM ne peut être programmée qu'une fois et ne peut être effacée. Les EPROM sont largement utilisées au sein des ordinateurs personnels car elles autorisent le fabricant à modifier le contenu des PROM avant que l'ordinateur soit réellement vendu. Cela signifie que les erreurs peuvent être fixées et des nouvelles versions des programmes installées rapidement avant livraison. A noter concernant la technologie des EPROM : les différents bits d'une EPROM sont programmés en injectant des électrons sous un voltage élevé sur la porte d'un transistor à effet de champ là où un bit à 0 est voulu. Les électrons capturés rendent le transistor passant, se lisant comme un 0. Pour effacer l'EPROM, on fournit aux électrons capturés assez d'énergie pour quitter la porte du transistor en bombardant le composant d'ultraviolets, au travers d'une fenêtre de quartz. Pour se prémunir d'un effacement lent sous l'effet de la lumière solaire ou des tubes fluorescents, cette fenêtre est couverte, en utilisation normale, d'un masque opaque.

- EEPROM : acronyme anglo-saxon pour *electrically erasable programmable read-only memory* (mémoire programmable et effaçable électriquement à accès en lecture seule). Prononcé double-e-prom ou e-e-prom, une EEPROM est un type particulier de PROM effaçable par exposition à une charge électrique. Comme les autres types de PROM, une EEPROM retient son contenu même lorsqu'elle est hors tension. Comme les autres types de ROM, une EEPROM est plus lente que la RAM. L'EEPROM est similaire à la mémoire flash (parfois appelée, flash-EEPROM). La principale différence est que les données d'une EEPROM doivent être écrites ou effacées octet par octet, alors que celles d'une mémoire flash le sont par blocs. Cela rend la mémoire flash plus rapide.

- FRAM : raccourci pour *Ferroelectric Random Access Memory* (mémoire à accès direct ferroélectrique), un type de mémoire non-volatile développé par Ramtron International Corporation. La FRAM combine les avantages de rapidité d'accès des DRAM et SRAM et la non-volatilité de la ROM. En raison de sa rapidité, elle remplace les EEPROM pour de nombreuses applications. Le terme FRAM lui-même est une marque déposée de Ramtron.

- NVRAM : abréviation de *Non-Volatile Random Access Memory* (mémoire non-volatile à accès direct), type de mémoire qui retient son contenu même hors tension. Un type de NVRAM est la SRAM qui est rendue non-volatile en la connectant sur une source électrique permanente comme une batterie. Un autre type de NVRAM utilise des EEPROM pour sauvegarder son contenu lors de la mise hors tension. Dans ce cas, la NVRAM est une combinaison de composants SRAM et EEPROM.

- Mémoire bulle : un type de mémoire non-volatile composé d'une fine couche d'un matériau qui peut être aisément magnétisé dans une seule direction. Lorsqu'un champ magnétique est appliqué sur une surface circulaire de cette substance qui n'est pas magnétisée dans la même direction, cette surface se réduit en un disque plus petit, ou bulle. On a cru que la mémoire bulle allait devenir la technologie phare des mémoires, mais elle n'a pas tenu toutes ses promesses. D'autres types de mémoires non-volatiles, comme les EEPROM, sont à la fois plus rapides et moins coûteuses que la mémoire bulle.

- Mémoire flash : un type spécifique d'EEPROM qui peut être effacé et reprogrammé par blocs. Le BIOS de nombreux PC récents est stocké sur une mémoire flash, et peut donc être mis à jour si nécessaire. Un tel BIOS est parfois appelé flash BIOS. La mémoire flash est également populaire dans les modems, car elle autorise le fabricant à prendre en compte de nouveaux protocoles au fur et à mesure des normalisation.

# B. Déterminer la capacité et la vitesse d'une EPROM à destination d'un adaptateur réseau

Cette section provient de la version 5.0 de la documentation du projet etherboot. Elle expose les méthodes permettant de déterminer la capacité et la vitesse de l'EPROM à utiliser pour un type donné d'adaptateur réseau.

La capacité minimale d'une EPROM, acceptée par une carte réseau est de 8 Ko (2764). Des tailles de 16 Ko (27128) ou 32 Ko (27256) sont la norme. Quelques cartes supportent des capacités de 64 Ko (27512). On peut voir aussi des dénominations portant un C après le 27, comme 27C256 ; cela signifie que l'EPROM est de type CMOS. Equivalente à la version sans C, elle constitue un bon choix eu égard à sa faible consommation électrique. Vous utiliserez la plus petite taille d'EPROM possible afin de ne pas utiliser plus que de besoin l'espace mémoire supérieur, car des extensions du BIOS peuvent en avoir besoin De même, vous voudrez ne pas acheter une EPROM trop chère. Les EPROM de 32 Ko et de 64 Ko semblent être les moins coûteuses à l'unité. Les EPROM de moindre capacité semblent plus chères car elles ne sont pas produites sur la chaîne principale de fabrication.

Si vous ne pouvez trouver dans la documentation la capacité de l'EPROM supportée par votre carte, pour des adaptateurs sur slot ISA uniquement, vous pouvez la découvrir par essais-erreurs. Les adaptateurs sur slot PCI n'activent pas l'EPROM sauf indication contraire du BIOS. Prenez une ROM contenant quelques données (par exemple un générateur de caractères) et branché là dans son emplacement. Prenez garde de ne pas utiliser une extension BIOS pour ce test car elle pourrait être détectée et vous empêcher de démarrer votre machine. En utilisant le programme DOS **debug**, effacer divers régions de la mémoire. Supposons que que vous découvrez les données dans la fenêtre mémoire située des adresses CC00:0 à CC00:3FFF (= 4000 en hexadécimal = 16384 en décimal). Cela indique qu'une EPROM de 16 Ko est nécessaire. Si vous découvrez qu'une région mémoire est dupliquée, mettons la région CC00:0 à CC00:1FFF identique à CC00:2000-CC0:3FFF, cela signifie que vous avez introduit une EPROM de 8 Ko dans un emplacement prévu pour une EPROM de 16 Ko, et qu'il vous faut essayer une EPROM de plus grande capacité.

Notez que parce que les EPROM à 28 broches bénéficient de la compatibilité ascendante, il est possible que vous utilisiez une EPROM de plus grande capacité dans une branchement prévu pour une plus petite. .

Si votre ROM est d'une capacité supérieure à la taille de l'image, par exemple, une ROM de 32 Ko contenant une image de 16 Ko, vous pouvez stocker l'image dans n'importe quelle moitié de la ROM. Vous pourrez lire des conseils stipulant de placer deux copies de l'image sur la ROM. Cela fonctionne mais n'est pas recommandé car la ROM risque d'être activée deux fois ou ne fonctionnera pas du tout si c'est une PCI/PnP. Cela est toléré par Etherboot car le code vérifie s'il a déjà été activé, et la seconde activation avorte. La méthode recommandée est de combler la seconde moitiée avec des données inertes. Remplir avec des 1 est recommandé car il s'agit là de l'état *naturel* de l'EPROM et nécessite moins de travail au programmateur de PROM. La commande Unix suivante engendre 16384 octets à la valeur 0xFF et les combine avec le contenu d'une ROM de 16 Ko pour créer une image de 32 Ko à l'usage du programmateur d'EPROM.

```
# (perl -e 'print "\xFF" x 16384'; cat bin32/3c509.lzrom) > 32kbimage
```

La vitesse d'EPROM nécessaire dépend de la façon dont elle est connectée au bus de l'ordinateur. Si l'EPROM est directement connectée sur le bus, cas de beaucoup de composants de type NE2000, il vous faudra probablement vous procurer une EPROM dont la vitesse est au moins égale à celle de la ROM du BIOS principal. Typiquement 120-150 ns. Certaines cartes réseau jouent les médiateurs vers l'EPROM, introduisant des états d'attente qui peuvent permettre l'utilisation d'une EPROM plus lente. La lenteur de l'EPROM n'a pas beaucoup d'incidence sur la vitesse d'exécution d'Etherboot car celui se duplique en mémoire avant de s'exécuter. Il parait que Netboot fait de même.

Si vous disposez de votre propre équipement de programmation d'EPROM, il existe une jolie collection d'utilitaires de conversion de formats de fichiers d'EPROM à http://www.canb.auug.org.au/~millerp/srecord.html. Les fichiers produits par le procédé de construction d'Etherboot sont entièrement en binaire. Un convertisseur simple de format, du binaire vers le format hexadécimal Intel, est disponible sur le site web d'Etherboot à http://etherboot.sourceforge.net/bin2intelhex.c. Vous pouvez aussi utiliser l'utilitaire objcopy, inclus dans le paquetage binutils :

```
# objcopy --input-target binary --output-target ihex binary.file intelhex.fi
# objcopy --input-target ihex --output-target binary intelhex.file binary.fi
```

Etherboot est considéré comme capable de créer des ROM compatibles avec les caractéristiques PnP des adaptateurs réseau sur slot PCI. Une erreur résiduelle dans les entêtes a été éliminée. Cependant, certains BIOS ne prennent pas en compte cette modification, aussi ai-je écris le script Perl swapdevids.pl pour effectuer la rotation des entêtes en cas de besoin. Vous devrez expérimenter les deux méthodes et déterminer laquelle fonctionne. Ou vous pouvez récupérer le contenu d'une ROM qui fonctionne (RPL, ou ROM PXE) en utilisant le script Perl disrom.pl. Les champs à examiner sont ceux concernant le type du périphérique (base, inférieur, interface). Cela doit être 02 00 00 mais certains BIOS demandent 00 00 02 à cause d'une ambiguité dans les spécifications originelles.

# C. Fournisseurs de stations sans disques

Le guide pratique original "Machines sans disque" mentionne les noms des fournisseurs suivants :

- Linux Systems Labs Inc., USA http://www.lsl.com. Cliquez sur "Shop on-line", puis sur "Hardware" où tous les modèles sont listés. Téléphone : 1-888-LINUX-88.
- Diskless Workstations Corporation, USA http://www.disklessworkstations.com.
- Unique Systems of Holland Inc., Ohio, USA http://www.uniqsys.com

# Références

[Diskless-HOWTO] *Guide pratique "Machines sans disque"*. http://www.linuxdoc.org/HOWTO/Diskless-HOWTO.html.

[Diskless-root-NFS-HOWTO] *Guide pratique "Système de fichier racine sur NFS pour machines sans disque"*. http://www.linuxdoc.org/HOWTO/Diskless-root-NFS-HOWTO.html.

[Bootdisk-HOWTO] *Guide pratique "Disque de démarrage"*. http://www.traduc.org/docs/HOWTO/vf/Bootdisk-HOWTO.html.

[ltsp] *linux terminal server project*. http://www.ltsp.org.

[plume] *plume*. http://plume.sourceforge.net.

[logilab] *Le site web de Logilab.org*. http://www.logilab.org.

[PowerUp2Bash] *Guide pratique "Du démarrage à la ligne de commande bash"*. http://www.traduc.org/docs/HOWTO/vf/From-PowerUp-to-bash-prompt-HOWTO.html.

[ThinClient] *Guide pratique "Clients légers"*. http://www.linuxdoc.org/HOWTO/Thin-Client-HOWTO.html.

[cdwriting] *Guide pratique "Gravage de CD"*. http://www.traduc.org/docs/HOWTO/vf/CD-Writing-HOWTO.html.

[etb] *Le projet etherboot*. http://etherboot.sourceforge.net.

[VMWare] *VMWare*. http://www.vmware.com.

[plex86] *plex86*. http://www.plex86.org.