PXE: configurer un serveur de démarrage PXE sous Linux

Table of Contents

Introduction

L'amorçage PXE permet à une station de travail de démarrer depuis le réseau en récupérant une image de système d'exploitation qui se trouve sur un serveur. L'image ainsi récupérée peut être le système d'exploitation brut ou bien le système d'exploitation personnalisé avec des composantes logicielles.

Cet article, présente les étapes de configuration d'unserveur de démarrage PXE à l'aide des services DNSMASQ et NFS.

Nous allons mettre en place un environnement de développement pour éviter d'avoir à manipuler sans cesse la microSD. Pour cela nous allons utiliser un serveur tftp pour que la carte récupère le noyau à partir du PC, et un montage nfs pour que la carte charge la rootfs directement à partir de notre dossier de travail.

Principes généraux

Aaprès une nouvelle installation, la taille totale de ces répertoires n'est pas très grande, allant de 30 à 40 Mo. La charge de fichiers principale existe dans les répertoires /usr et /opt. Il est donc possible de créer un Répertoire de chaque client sans disque contenant les répertoires susmentionnés et des points de montage pour des répertoires tels que/usr qui seront exportés par le serveur. Le processus de démarrage, comme supposé par ce document, est le suivant:

- Lorsqu'un ordinateur utilise l'environnement PXE (Pre-eXecution Environment) pour démarrer directement à partir du réseau, il doit obtenir une adresse IP d'un serveur DHCP.
- Le serveur DHCP peut également lui donner les détails d’un serveur TFTP à partir duquel extraire un fichier exécutable, généralement appelé pxelinux.0 pour un client Linux (DNSMASQ, s’occupe de pointer les systèmes d’amorçage sur le serveur TFTP en fournissant l’option enable-tftp dans le fichier de configuration dnsmasq et il n'y a donc pas besoin d’un serveur DHCP distinct).
- Une fois que le client a récupéré et exécuté pxelinux.0, il est codé en dur pour rechercher un fichier dans le sous-répertoire pxelinux.cfg / relatif à l'emplacement où pxelinux.0 a été trouvé.
- Le noyau prend le contrôle du système, identifie les périphériques du système et utilise DHCP pour obtenir l'adresse IP correspondant à l'adresse matérielle de la carte réseau.
- Avant de passer à un niveau d'exécution, il appelle un script décrit dans le fichier /etc/inittab, qui est chargé de la création du cache de la bibliothèque, de l'initialisation et du montage d'un fichier d'échange, du chargement de certains modules du noyau et Définit le nom d'hôte.
- Le script de démarrage se termine et le programme init passe au runlevel specifie: il commence à exécuter les scripts situés dans le répertoire /etc/rc.d/rcX, où 'X' correspond au nom du niveau d'exécution. Ces scripts sont chargés de démarrer le mappeur de ports et de monter le NFS exporté /usr, /home Et les répertoires /opt.
- L'utilisateur peut se connecter.

En résumé, pour démarrer un système linux il faut:

- un noyau (vmlinuz: noyau au format compressé),
- un rootfs (initrd: fichier e permettant d'utiliser des fonctionalités non incluses dans le noyau lors du chargement de celui-ci, comme le support d'un filesystem compilé en module, ou le driver d'un matériel spécifique compilé en module par exemple).

les tâches à effectuer pour mettre en oeuvre un serveur PXE sont donc les suivantes:

- Préparer le serveur PXE et créer les répertoires spécifiques
- Préparer un système de base à exporter vers les ĥôtes
- Préparer un noyau vmlinuz compatible
- Préparer le serveur pour exporter certains répertoires et démarrez un service bootp.

Préparation du serveur PXE et des répertoires spécifiques

Installation

Installer DNSMASQ (ou un serveur TFTP et un serveur DHCP) la prise en charge de NFS.

- Pour Debian/Ubuntu

```
sudo apt-get update
sudo apt-get install dnsmasq nfs-kernel-server
```

- Pour Centos/RHEL

```
yum -y install dnsmasq nfs-utils
```

Créer le répertoire racine du serveur

```
mkdir /tftpboot
```

Configuration de dnsmasq

Créer le fichier de configuration `/usr/local/etc/dnsmasq.conf` avec le contenu suivant (ceci activera le serveur DHCP et le serveur proxy DHCP requis pour attribuer l'adresse IP des clients PXE):

```
cat > /usr/local/etc/dnsmasq.conf <<EOF
interface=bce1
dhcp-range=192.168.1.80,192.168.1.85
pxe-service=x86PC, "pxelinux", pxelinux
enable-tftp
```

```
tftp-root=/tftpboot
dhcp-boot=pxelinux.0,servername,192.168.0.50
EOF
```

Et démarrer le service

```
sysrc dnsmasq_enable=yes
service dnsmasq start
```

Configuration de RARP

Pour permettre aux clients de booter de façon automatique, si on ne veut pas utiliser DHCP, il fuadra configurer rarp.

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

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

pour connaître l'adresse de la carte Ethernet, booter sur le client un noyau avec le support de la carte Ethernet.

Il peut être judicieux d'inclure cette ligne dans `/etc/rc.d/rc.sysconfig` pour ne pas avoir à taper la commande à chaque boot du serveur.

Configuration du serveur NFS

Après l'installation, vous devrez créer un partage NFS, un répertoire pouvant être installé à partir de clients distants. Il est recommandé de ne pas utiliser le répertoire de partage NFS à d'autres fins.

Créer le répertoire NFS à partager avec La commande

```
mkdir /tftpboot/nfs
```

Après avoir créé le répertoire, éditer le fichier /etc/exports.

Ce fichier contient une liste des partages NFS, les adresses IP à partir desquelles ils peuvent être montés, le type d'accès (en lecture seule ou en lecture-écriture) et d'autres options de partage. Supposons que votre sous-réseau IP avec les serveurs pour Bare-Metal Restore soit 192.168.5.0/24 Ajouter à /etc/exports la ligne suivante:

```
/tftpboot/nfs 192.168.5.0/24(ro,no_root_squash,no_subtree_check)
```

Enregistrer le fichier et exécuter la commande suivante:

```
exportfs -r
```

Ensuite, démarrer le service NFS en exécutant la commande suivante:

```
/etc/init.d/nfs start
```

Pour s'assurer que le partage NFS est visible pour le client, exécuter la commande suivante sur le serveur NFS:

```
showmount -e
```

La commande devrait retourner quelque chose comme ceci:

```
Export list for pxeserver:
/tftpboot/nfs 192.168.5.0/24
```

Note: Le service NFS dépend du service `portmap` et ne fonctionnera généralement pas si `portmap` n'est pas en cours d'exécution, mais cette dépendance n'est pas toujours codée en dur dans le script de démarrage, aussi le script essaie-t-il parfois de démarrer le service NFS sans chercher à savoir si portmap est déjà démarré Et si c'est le cas, dans ce cas, le démarrage de NFS échouera. Si, pour quelque raison que ce soit, le service NFS ne parvient pas à démarrer, essayer tout d'abord de lancer portmap en exécutant la commande `/etc/init.d/portmap start`

Afin d'optimiser NFS:
* Réduire la taille de la fenêtre TCP (paramètre **wsize** pour Linux) à la valeur la plus proche du MTU de votre type de réseau. Pour Ethernet, une bonne valeur de **wsize** est 2048 octets tant que le MTU est de 1536 octets. C'est généralement une bonne idée car La charge de trafic principale entre les clients et le serveur est constituée de petits paquets et, dans le cas de grands programmes tels que X ou StarOffice, il existe un grand nombre de paquets fragmentés, ce qui peut varier dans votre cas, en fonction des besoins des utilisateurs. Vos utilisateurs
* Pour une installation volumineuse, diviser l'espace réservé aux postes de travail en deux disques SCSI ou plus, afin de permettre des écritures et des lectures conséquentes sur les deux disques, ce qui augmentera le temps de réponse et réduirait le temps de latence avant la fin de la demande.

Configuration de base PXE

Cette étape consiste à télécharger et copier tous les fichiers de démarrage PXE dans le répertoire du serveur TFTP (/tftpboot) et à créer les fichiers et les répertoires nécessaires.

- **Méthode 1**: télécharger les binaires et les copier dans le répertoire `/tftpboot`:

```
cd /tftpboot
fetch https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.zip
unzip -d syslinux syslinux-6.03.zip
cp syslinux/bios/memdisk/memdisk /tftpboot
cp syslinux/bios/core/pxelinux.0 /tftpboot
cp syslinux/bios/com32/elflink/ldlinux/ldlinux.c32 /tftpboot
cp syslinux/bios/com32/menu/menu.c32 /tftpboot
cp syslinux/bios/com32/libutil/libutil.c32 /tftpboot
cp syslinux/bios/com32/modules/pxechn.c32 /tftpboot
cp syslinux/bios/com32/lib/libcom32.c32 /tftpboot
cp syslinux/bios/com32/chain/chain.c32 /tftpboot
cp syslinux/bios/com32/modules/reboot.c32 /tftpboot/
rm syslinux-6.03.zip
rm -rf syslinux
```

Le fichier pxelinux.0 et le menu associé (menu.c32) doivent être COPIÉS (plutôt que liés de manière logicielle) du dossier `/usr/lib/syslinux` dans le dossier /tftpboot).

- **Méthode 2**: Installer le paquet syslinux de la disrtibution et copier les fichiers de démarrage PXE dans le répertoire `/tftpboot`:

```
# Pour Debian/Ubuntu
sudo apt-get install syslinux-common

# Pour Centos/RHEL
yum -y install syslinux

cp -r /usr/lib/syslinux/* /tftpboot/
```

Créer un fichier `/tftpboot/pxelinux.cfg/default` coder l'image du noyau et l'argument de démarrage dans la configuration "par défaut".

```
mkdir /tftpboot/pxelinux.cfg
cat > /tftpboot/pxelinux.cfg/default <<EOF
SERIAL 0 115200
CONSOLE 0
UI menu.c32
TIMEOUT 300
MENU TITLE PXE BOOT MENU
LABEL Test
  MENU LABEL Test
```

```
LABEL reboot
  MENU LABEL reboot
  KERNEL reboot.c32
EOF
```

Préparation d'un noyau vmlinuz compatible

Le noyau du client doit avoir les options NFS-Root, Rarp configurées, de la configuration de la carte Ethernet, et éventuellement des périphériques SCSI ou autres.

Création du noyau

Télécharger les sources du noyau

Récupérer le dépôt git de Linus,

```
git: //git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

Choisir toujours le noyau le plus récent à compiler. Construire un noyau plus ancien ne peut que vous faire gagner du temps à la mise à jour des programmes nécessaires.

Configurer le noyau pour NFS

Le fichier de configuration `/arch/x86/configs/x86_64_defconfig` est déjà assez complet. Il faut ajouter les configurations suivantes pour créer `x86_64_nfs_defconfig` pour la fonctionnalité rootfs NFS:

```
CONFIG_NETWORK_FILESYSTEMS=y
CONFIG_ROOT_NFS=y
CONFIG_NFS_USE_KERNEL_DNS=y
# CONFIG_NFSD is not set
CONFIG_NFS_COMMON=y
CONFIG_E1000E=y
```

- Les paramètre `CONFIG_NFS` sont substitués au paramètre `CONFIG_NFSD=m` défini précédemment dans le fichier.
    - `CONFIG_E1000E` (driver PCI-Express Intel(R) PRO/1000 gigabit ethernet) n'est pas techniquement une configuration NFS rootfs, mais le noyau a besoin d'un pilote de périphérique compilé de manière statique pour pouvoir obtenir une communication IP avant que le rootfs ne soit disponible.

Facultatif mais recommandé):

- **CONFIG_FRAME_POINTER=y** - sauvegarde les informations du cadre, pour permettre à gdb de construire plus efficacement la trace de pile
- **CONFIG_KGDB_SERIAL_CONSOLE=y** - La kgdb sur Ethernet n'est pas dans la ligne principale, restez donc fidèle à la réalité, cela permet également le kgdbwait et un débogage précoce
- **# CONFIG_DEBUG_RODATA is not set** - Cette option marque certaines régions de l'espace mémoire du noyau en tant que RO.
- **# CONFIG_CC_OPTIMIZE_FOR_SIZE is not set** - lorsqu'on dispose de beaucoup d'espace disque mais que le processeur est relativement lent.
- Pour ajouter ajouter la fonctionnalité `kgdb`, utiliser les options suivantes:

```
CONFIG_KGDB=y
CONFIG_DEBUG_INFO=y
```

- La structure ftrace fournit aux utilisateurs plusieurs fonctionnalités de traçage,qui peuvent être utilisés pour analyser le temps de latence de réveil, les commutateurs de tâches, les événements du noyau, etc. Pour ajouter les configurations suivantes pour la fonctionnalité `ftrace`, utiliser les options suivantes:

```
CONFIG_KPROBES_ON_FTRACE=y
CONFIG_HAVE_KPROBES_ON_FTRACE=y
# CONFIG_PSTORE_FTRACE is not set
CONFIG_DYNAMIC_FTRACE=y
CONFIG_HAVE_DYNAMIC_FTRACE=y
CONFIG_HAVE_DYNAMIC_FTRACE_WITH_REGS=y
CONFIG_HAVE_FTRACE_MCOUNT_RECORD=y
CONFIG_FTRACE=y
CONFIG_FTRACE_SYSCALLS=y
CONFIG_DYNAMIC_FTRACE=y
CONFIG_DYNAMIC_FTRACE_WITH_REGS=y
CONFIG_FTRACE_MCOUNT_RECORD=y
# CONFIG_FTRACE_STARTUP_TEST is not set
```

Lancer `make Menuconfig`, le noyau peut être construit en fonction des besoins spécifiques en pilotes, mais il **DOIT** contenir les options suivantes:

- Prise en charge intégrée de la carte réseau du client (`Network device support ---> Select your card driver`).
- Prise en charge intégrée du protocole BOOTP (`IP: kernel level autoconfiguration ---> IP: BOOTP support`).
- Support intégré pour NFS et racine sur NFS (`File systems ---> Network File Systems ---> NFS file system support and File systems ---> Network File Systems ---> NFS file system support ---> Root over NFS`).
- Prise en charge intégrée des périphériques en boucle (`Block devices ---> Loopback device support`).
- Prise en charge du pilote de framebuffer VESA

Construire le noyau

```
cp ~/kernel.config ~/linux/.config
cd ~/linux
make olddefconfig
make clean
make -j 4
```

Mettre à disposition le noyau

Après que “make” soit passé, l’image compressée du noyau et les rootfs tarés sont dans `/usr/src/linux/arch/i386/boot`.

```
ls -gh sortie/images /
Total 25M
-rw-rw-r-- 1 Henry 5,4 M 28 novembre 18:07 bzImage
-rw-rw-r-- 1 henry 19 mois 28 novembre 18:19 rootfs.tar
```

Copier le noyau sur le serveur

Copier le noyau dans le répertoire `/tftpboot` ainsi que tous les modules créés dans le répertoire base `/lib/modules`:

```
cp arch/i386/boot/bzImage /tftpboot/vmlinuz
```

Définition du périphérique racine

Le serveur doit exposer certaines parties de son système de fichiers pour que le client puisse les monter et les utiliser comme système de fichiers racine. En l'absence de bootloader les valeurs de périphérique racine codées dans le noyau avec la commande `rdev` seront prises en compte.

Dans `/tftpboot/pxelinux.cfg/default` le noyau doit être invité à utiliser le périphérique `/dev/nfs` pour monter son système de fichiers (par ex.: `append root=/dev/nfs ip=dhcp init=/sbin/init`). Il faut donc créer dans le node `/dev/nfs` et sert d’espace réservé.

```
mknod /dev/nfs c 0 255
```

Puis indiquer au noyau d'utiliser ce périphérique en tant que système de fichiers racine à l'aide de la commande rdev.

```
rdev /tftpboot/vmlinuz /dev/nfs
```

Préparation du système de base à exporter vers les clients

Cette section aborde les différentes méthodes pour mettre à disposition du client :

- soit un environnement NFS-Root
- soit un initramfs

initrd vs initramfs

Le disque root initial

initrd est une structure de disque en mémoire (ramdisk) qui contient les outils et scripts nécessaires au montage des systèmes de fichiers requis avant que le contrôle ne soit passé à l'application init du système de fichier root. Le noyau Linux déclenche le script de configuration (ordinairement appelé linuxrc mais ça n'est pas une obligation) sur ce disque root, lequel prépare le système, commute vers le système de fichiers root réel et appelle ensuite init.

Bien que la méthode de l'initrd soit suffisante, elle présente quelques inconvénients :

- C'est un périphérique de blocs à part entière, qui nécessite le surcoût d'un système de fichiers complet sur lui, et a une taille fixe. Choisissez un initrd trop petit, et vous ne pourrez y placer tous les scripts nécessaires. Choisissez-le trop grand et vous allez gaspiller la mémoire.
- Comme c'est un périphérique réel statique, il consomme de la mémoire cache dans le noyau Linux et est sujet aux méthodes de gestion de fichiers utiliser(tel que paging) accroissant la consommation de mémoire de initrd.

C'est pour résoudre ces problèmes, que le système de fichiers virtuel initial (initramfs) a été inventé.

Le système de fichiers virtuel initial

Un système de fichiers virtuel initial (initramfs) est un système de fichiers initial en mémoire ram basé sur tmpfs (un système de fichiers léger de taille flexible, en mémoire), qui n'utilise pas un périphérique de blocs séparé (ainsi aucun cache n'est nécessaire, ce qui élimine les surcoûts mentionnés précédemment). Tout comme le initrd, il contient les outils et les scripts nécessaire au montage des systèmes de fichiers avant que le contrôle e soit passé à l'application init du système de fichiers root réel.Ces outils peuvent être, des couches d'abstraction du chiffrage (pour les système chiffrés/cryptés), des gestionnaire de volumes logiques, le raid logiciel, des chargeurs de systèmes de fichiers basés sur des pilotes bluetooth, etc.

Le contenu de l'initramfs est créé en créant une archive cpio. cpio est une ancienne (mais éprouvé) solution d'archivage de fichiers. Les archives produites sont appelées fichiers cpio. cpio est comparable à l'archiveur tar. cpio a été choisi ici parce que le code était plus facile à mettre en œuvre et qu'il prend en charge des fichiers de périphériques, ce que tar ne pouvait pas.

Tous les fichiers, les outils, les bibliothèques, les fichiers de configuration (si applicables), etc. sont placés dans l'archive cpio. L'archive est ensuite compressée avec l'utilitaire gzip et stockée avec le noyau Linux. Le chargeur d'amorçage (boot loader) le présente ensuite au noyau Linux au moment du démarrage ainsi le noyau sait qu'un système de fichiers virtuel initial (initramfs) est nécessaire.

Une fois détecté, le noyau Linux crée un système de fichiers tmpfs, extrait le contenu de l'archive dans ce système de fichiers, et lance le script init situé à la racine du système de fichiers tmpfs. Ce script monte ensuite la racine (root) du système de fichiers réel (après s'être assuré qu'il peut le faire, par exemple en chargeant des modules additionnels, en préparant un couche d'abstraction du chiffrage, etc.) ainsi que les autres systèmes de fichiers vitaux (comme /usr and /var).

Dès lors que le système de fichier racine (root) et les autres systèmes de fichiers vitaux sont montés, le script init du initramfs commute la racine (root) vers le système de fichiers racine réel et, finalement, appelle le binaire /sbin/init sur ce système pour continuer le processus de démarrage.

Préparation d'un environnement NFS-Root

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

Le principe est de créer sur le serveur toute l'arborescence nécessaire au fonctionnement du client.

Une solution consiste à recopier la racine du serveur puis de modifier les fichiers spécifiques. Cette solution est extrêmement gourmande en place surtout si l'on envisage plusieurs clients utilisant NFS-Root.

La solution décrite ici permet de limiter le nombre de fichiers à recopier en utilisant directement ceux disponibles sur le serveur.

Création de la partition root du client

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

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

Pour pouvoir démarrer un système Linux, il faut lui fournir les éléments suivants:

- **Le répertoire /sbin**, qui contient le programme init, qui est chargé de lancer d'autres programmes et des scripts de démarrage au cours du processus de démarrage, ainsi que les scripts de démarrage dans le cas de SuSE, ainsi que des programmes utiles dans le programme

portmap. Et de nombreux autres programmes nécessaires avant de monter le répertoire /usr.
- **Le répertoire /lib**, qui contient les bibliothèques libc indispensables si votre init est lié dynamiquement.
- **Le répertoire /bin**, qui contient les commandes de fichier et les shells permettant d'exécuter des scripts de démarrage.
- **Le répertoire /etc**, qui contient les fichiers de configuration pour la plupart des programmes et les répertoires rc.d qui sont les répertoires par défaut des scripts de démarrage.
- **Le répertoire /var** est une zone de spool pour les programmes qui veulent écrire quelque part, il est divisé en plusieurs sous-répertoires avec une facilité d'utilisation alternative.
- **Le répertoire /dev**, qui contient des périphériques spéciaux permettant aux programmes de communiquer avec les périphériques de l'ordinateur via le noyau.

Ainsi que les autres répertoires dont on aura besoin: home, mnt, proc, common, tmp, usr.

Dans le répertoire `/tftpboot/client-IP`, il faut créer les répertoires puis les remplir.

Les répertoires bin, dev, sbin et lib

- **bin** doit contenir les programmes mount, sh (qui est généralement un lien vers bash qu'il faut donc également copier, sinon recopier le shell pointé par sh)
- **dev** doit contenir l'ensemble des accès aux périphériques (les devices console, mouse, cdrom et modem sont des liens symboliques, à modifier le cas échéant). Pour le générer, utiliser : `cp -a /dev /tftpboot/client-IP/dev`
- **sbin** doit contenir le programme init
- **lib** doit avoir quelques librairies indispensables, à recopier avec : `cp -a /lib/libc.so* /lib/libtermcap.so* /lib/ld* /tftpboot/client-IP/lib`

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

Les répertoires var et etc, partagés entre les clients

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

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

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

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

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

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

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

Paramétrage des fichiers etc des clients

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

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

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

`xx.xx.xx.xx:` étant bien entendu à remplacer par l'adresse ip serveur NFS-Root.

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

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

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

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

autres lignes faisant un mount ou un umount.

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

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

Préparation d'une image inird

initrd (INITial RamDisk) est une image d'un système d'exploitation minimal initialisé au démarrage du système. Fichier identifié sous le nom initrd.img-<version> présent dans /boot

Cette partie est très spécifique à la distribution et que certaines des choses décrites ici ne sont pas applicables au cas particulier.

Pour préparer le système de base, il faut installer une distribution préférée puis installer tous les programmes que l'on veut mettre à la disposition des utilisateurs puis configurer le nouveau système.

Installation de debootstrap

Debootstrap permet l'installation d'un système dans une racine. Il exige plusieurs paramètres comme la racine d'installation, l'architecture matérielle, la distribution et l'archive FTP ou HTTP Debian à solliciter pour le téléchargement.

Debootstrap accepte aussi en entrée une liste d'archives, une liste de paquets à inclure et une liste de paquets à exclure. On installera un ensemble d'outils qu'il semble indispensable d'inclure dès le début (par exemple le noyau, des firmwares pour un support étendu de tous les matériels et un ensemble d'outils d'audit).

Sous Debian/Ubuntu

```
apt-get install debootstrap
```

Sous CentOS/ RHEL

Le paquet debootstrap n'est pas disponible dans un référentiel CentOS standard. Pour le rendre disponible, il faut d’abord activer le référentiel EPEL. Télécharger le package de référentiel EPEL:

```
# wget
```

```
http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
```

Et installer le paquet en utilisant la commande RPM:

```
# rpm -Uvh epel-release-7-5.noarch.rpm
```

Le référentiel EPEL doit maintenant être activé.

```
# yum repolist | grep epel
  * epel: epel.mirror.digitalpacific.com.au
```

Maintenant, installer simplement debootstrap en utilisant yum:

```
# yum install debootstrap.noarch
================================================================================
====
 Package           Architecture Version                     Dépôt
Taille
================================================================================
====
Installation :
 debootstrap       noarch       1.0.109-2.el7               epel
84 k
Installation pour dépendances :
 dpkg              x86_64       1.18.25-8.el7               epel
1.3 M
Mise à jour pour dépendances :
 perl              x86_64       4:5.16.3-294.el7_6          updates
8.0 M
 perl-libs         x86_64       4:5.16.3-294.el7_6          updates
688 k

Résumé de la transaction
================================================================================
====
Installation   1 Paquet (+1 Paquet en dépendance)
Mettre à jour           ( 2 Paquets en dépendance)
```

Définition des variables d'environnement

Par commodité on va définir:

- une variable d'environnement correspondant à la racine du système `$SIDUS`
- un alias `sidus` permettant l'exécution d'une commande par chroot avec une option particulière d'installation de paquet.

Pour une architecture x84_64 ou AMD64

```
export SIDUS=/tftpboot/nfs/jessie64nfs
alias sidus="DEBIAN_FRONTEND=noninteractive chroot ${SIDUS} $@"
ARCH=amd64
```

Pour une architecture i386

```
export SIDUS=/tftpboot/nfs/jessie32nfs
alias sidus="DEBIAN_FRONTEND=noninteractive setarch i686 chroot ${SIDUS} $@"
ARCH=i386
```

Insallation du système

Lancement de l'installation

```
debootstrap --arch $ARCH --components='main,contrib,non-free' jessie $SIDUS
http://ftp.debian.org/debian
```

A la suite de cette commande, on doit prendre quelques précautions :

- normalement, si le paquet Debian est un service, ce dernier démarre après son installation. Il faut donc inhiber le lancement de ce service par la définition d'un hook (/usr/sbin/policy-rc.d) :

```
#!/bin/sh
exit 101
```

Ou sous forme de commande :

```
printf '#!/bin/sh\nexit 101\n' > ${SIDUS}/usr/sbin/policy-rc.d
chmod +x ${SIDUS}/usr/sbin/policy-rc.d
```

- des paquets exigent l'accès à la liste des processus, du système, des périphériques, de la mémoire virtuelle, des pointeurs de périphériques. Il faut donc “binder” le montage de ces dossiers du système hôte au système SIDUS :

```
sidus mount -t proc none /proc
sidus mount -t sysfs sys /sys
mount --bind /dev/pts ${SIDUS}/dev/pts
```

Définition du mot de passe root

```

```

```
echo "root:MyStrongPassword" | sidus chpasswd
```

Installation de paquets jugés nécessaires

```
sidus apt-get update
sidus apt-get -y install aptitude dselect dracut dracut-network isc-dhcp-common isc-dhcp-client openssh-server locales aufs-tools firmware-linux-nonfree bridge-utils firmware-linux firmware-bnx2 dstat sysstat iftop htop iotop emacs lsof tshark mbw strace memtest86 cpuburn dbench iozone3 psmisc console-setup less vim unscd nfs-common mlocate tshark linux-image-${ARCH} linux-headers-${ARCH}
```

Presque toutes les distributions démarrent ces services:

- **Inetd**: le superdeamon Internet responsable du démarrage d'autres démons tels que telnet, ftp, etc.
- **Syslogd**: le démon de journalisation, inutile sur un client sans disque, car toutes les modifications sont apportées aux fichiers facilement remplaçables.
- **Httpd**: le serveur Web Apache, inutile pour des raisons évidentes.
- **Dhcpclient**: Nécessaire pour l'acquisition automatique d'une adresse IP. Dans le cas contraire, cela est fait par le noyau.
- **Lpd**: le démon de l'imprimante par ligne, n'est nécessaire que si vous avez une imprimante connectée à un hôte. Dans la plupart des cas, cela n'est pas nécessaire.

De plus, selon l'installation, il se peut que des services sshd, nscd, cupsd et autres services réseau démarrés ne soient pas nécessaires sur les clients. Pour désactiver ces services, supprimer leurs entrées du répertoire d'exécution sous `/etc/rc.d/X`.

Désinstaller ensuite tous ces services du système de base.Le seul service qui paraisse raisonnable est le démon de mise en cache NameServer, qui permet de réduire considérablement le trafic réseau

Configuration du script inittab

Après le démarrage `init` exécute un script décrit dans `/etc/inittab`. Ce script a un travail très spécifique à effectuer: Amener le système dans un état permettant le démarrage d'autres programmes. Dans la plupart des distributions, ce script permet de:

- Monter les systèmes de fichiers /proc,/dev/pts et swap.
- Activer les matrices RAID et fscks le système de fichiers racine.
- Ajuster l'horloge.
- Démarrer le démon du noyau pour le chargement automatique des modules.
- Exécuter les scripts client définis par l'utilisateur.
- Définir certains paramètres du noyau.

Pour demarrer un hôte sur le Net, il faut apporter les modifications suivantes à ce script:

- Supprimer toutes les entrées qui effectuent `fsck` ou initialisent des aires de raid et ajouter en haut du script cette commande: `mount -o remount, rw /` car le client doit disposer d'un accès rw au répertoire racine lorsqu'il s'amorce.

- Ne pas laisser le démon du noyau démarrer tant que toutes les partitions ne sont pas montées
- Monter une partition de swap.
    - Démarrer le portmapper Si le système dispose d'un répertoire spécifique pour le démarrage des scripts de démarrage, y placer le script de démarrage portmapper en lui donnant la priorité la plus élevée possible, par exemple: `ln -s /etc/rc.d/portmap /etc/rc.d/`.
    - Placer le script de montage du système de fichiers NFS dans le répertoire spécifique du système pour les scripts de démarrage dont la priorité est inférieure à celle du gestionnaire de ports, par exemple, ln -s /etc/rc.d/nfs /etc/rc.d/boot/S02nfs pour SuSE.
    - Supprimer toutes les entrées qui montent automatiquement les partitions locales et toutes les entrées qui lancent un démon automounter pour RedHat.

Configuration d'une partition de swap

L'échange sur NFS n'est pas autorisé par le noyau et ne fonctionne pas non plus. Vous ne pouvez pas utiliser swapon sur des fichiers qui se trouvent sur un système de fichiers NFS monté. Nous devons faire quelques astuces pour l'activer:

- Créer le fichier d'échange. Sa taille peut être variable, mais pour une machine disposant de 128 Mo de RAM, une taille d'échange de 40 à 50 Mo semble raisonnable. La commande permettant de créer le fichier d'échange est la suivante: `dd if=/dev/zero of=/var/swap bs=1k count=Xk` où X représente le nombre de Mo que votre swap devrait être Il est également nécessaire de placer le fichier d'échange sous /var tant qu'il est monté au démarrage.
- Formater le fichier d'échange à l'aide de la commande mkswapfs.
- Initialiser un périphérique en boucle à l'aide du fichier d'échange. La commande est la suivante: `losetup /dev/loop0/var/swap`.
- Monter le périphérique en boucle avec la commande `mount /dev/loop0 swap`.

Il faut initialiser une partition de swap au tout début du processus de démarrage. Il faut donc placer les commandes 2 à 4 quelque part vers le haut du script de démarrage. La première commande prend beaucoup de temps, en particulier dans le cas d'un réseau chargé. Copiez un fichier d'échange dans le système de base et ne le supprimez pas lorsque vous créez des répertoires pour chaque hôte.

Modification de /etc/fstab

Le fichier `/etc/fstab` contient des entrées pour le montage automatique des systèmes de fichiers au démarrage. Dans notre cas, nous devons placer les lignes suivantes à la fin:

```
server_IP:/usr/local/linux/base/usr /usr nfs nfsvers=3,wsize=2048,tcp 0  0
server_IP:/usr/local/linux/base/opt /opt nfs nfsvers=3,wsize=2048,tcp 0  0
server_IP:/usr/local/linux/base/lib/modules /lib/modules nfs nfsvers=3
wsize=2048,tcp 0  0
fileserver_IP:/home /home nfs nfsvers=3,wsize=2048,tcp 0  0
```

Commenter les lignes qui montent des partitions locales.

Paramétrage de la localisation

Paramétrage de la localisation, la langue, le fuseau horaire à partir du serveur lui même :

```
mv ${SIDUS}/etc/locale.gen ${SIDUS}/etc/locale.gen.orig
mv ${SIDUS}/etc/timezone ${SIDUS}/etc/timezone.orig
mv ${SIDUS}/etc/default/keyboard ${SIDUS}/etc/default/keyboard.orig
cp /etc/locale.gen ${SIDUS}/etc/locale.gen
cp /etc/timezone ${SIDUS}/etc/timezone
cp /etc/default/keyboard ${SIDUS}/etc/default/keyboard
sidus locale-gen
sidus dpkg-reconfigure tzdata
```

Installation de l'environnement graphique

Installation de l'environnement graphique de poste de travail et d'un environnement plus léger

```
sidus tasksel install desktop
sidus tasksel install xfce-desktop
```

Purger des paquets pour une utilisation dans XFCE

```
sidus apt-get purge -y network-manager modemmanager gdm3 gnome-session
gnome-terminal nano vim-tiny
sidus apt-get clean
```

Paramétrage d'un boot par NFS utilisant Dracut

- **dracut** est utilisé pour créer une image initramfs en copiant des outils et des fichiers d'un système installé et en le combinant avec le framework dracut (infrastructure initramfs pilotée par événement), qui se trouve généralement dans `/usr/lib/dracut/modules.d`
- **AuFS** (Advanced multi layered Unification FileSystem or AnotherUnionFS) est un service du système de fichiers de Linux dérivé de Unionfs, qui permet de fusionner plusieurs points de montage appelés « branches » : c'est un union mount.

```
mv ${SIDUS}/usr/lib/dracut/modules.d/90aufs/aufs-mount.sh
${SIDUS}/root/aufs-mount.sh.orig
cat > ${SIDUS}/usr/lib/dracut/modules.d/90aufs/aufs-mount.sh <<EOF

#!/bin/sh

# rend un nfsroot en lecture seule accessible en écriture avec aufs
# le nfsroot est déjà monté sur $NEWROOT
# ajoute le paramètre aufs au noyau, pour activer cette fonctionnalité
```

```
. /lib/dracut-lib.sh

aufs=$(getargs aufs)

if [ -z "$aufs" ] ; then
    return
fi

modprobe aufs

# a little bit tuning
mount -o remount,nolock,noatime $NEWROOT

mkdir -p /.sidus/ro
mount --make-runbindable /
mount --move $NEWROOT /.sidus/ro

mkdir -p /.sidus/rw
mount -o uid=0,mode=0755 -t tmpfs none /.sidus/rw

mount -t aufs -o noxino,br=/.sidus/rw:/.sidus/ro=rr aufs $NEWROOT

mkdir -p $NEWROOT/.sidus/rw
mkdir -p $NEWROOT/.sidus/ro
mount --move /.sidus/rw $NEWROOT/.sidus/rw
mount --move /.sidus/ro $NEWROOT/.sidus/ro
EOF

chmod 755 ${SIDUS}/usr/lib/dracut/modules.d/90aufs/aufs-mount.sh
```

Modifier le bail DHCP

```
sed -i "s/=\$new_dhcp_lease_time/=forever/"
${SIDUS}/usr/lib/dracut/modules.d/40network/dhclient-script.sh
```

Créer le démarreur à partir de Dracut

```
sidus dpkg-reconfigure dracut
```

Suppression du hostname pour le paramétrage automatique du HOST

sidus rm /etc/hostname

Définition de l'interface de loopback

```
tee ${SIDUS}/etc/network/interfaces <<EOF
```

```
auto lo
iface lo inet loopback
EOF
```

Pour la résolution DNS

```
cp /etc/resolv.conf ${SIDUS}/etc/resolv.conf
```

Tuning système associé aux interfaces réseau

```
wget -O ${SIDUS}/etc/sysctl.d/gc.conf

net.ipv4.neigh.default.gc_thresh1 = 1024
net.ipv4.neigh.default.gc_thresh2 = 4096
net.ipv4.neigh.default.gc_thresh3 = 8192
net.ipv4.neigh.default.gc_stale_time = 86400
net.ipv4.neigh.eth0.gc_stale_time = 86400
net.ipv4.neigh.eth1.gc_stale_time = 86400
net.ipv4.neigh.default.gc_interval = 86400
```

Contrôle des ressources systèmes (ulimit)

inux permet de borner les ressources allouées aux utilisateurs, ou aux groupes d'utilisateurs, via le fichier « /etc/security/limits.conf »

```
mv ${SIDUS}/etc/security/limits.conf ${SIDUS}/etc/security/limits.conf.orig
wget -O ${SIDUS}/etc/security/limits.conf

# /etc/security/limits.conf
#
Chaque ligne décrit une limite pour un utilisateur sous la forme:
#
#domain>        <type>  <item>  <value>
#
#Où:
#<domain> peut être:
# - un nom d'utilisateur
# - un nom de groupe, avec la syntaxe @group
# - le caractère générique *, pour l'entrée par défaut
# - le caractère générique %, peut également être utilisé avec la syntaxe
%group,
# pour limite maxlogin
# les limites de groupe et génériques ne sont pas appliquées à la racine.
# Pour appliquer une limite à l'utilisateur root, <domain> doit être
# le nom d'utilisateur littéral root.
#
```

```
# <type> peut avoir les deux valeurs:
# - "soft" pour faire respecter les limites souples
# - "hard" pour imposer des limites strictes
#
# <item> peut être l'un des suivants:
# - core - limite la taille du fichier core (Ko)
# - data - taille maximale des données (Ko)
# - fsize - taille maximale du fichier (Ko)
# - memlock - espace d'adressage maximum en mémoire verrouillée (Ko)
# - nofile - nombre maximal de fichiers ouverts
# - rss - taille maximale du groupe de résidents (Ko)
# - stack - taille maximale de la pile (Ko)
# - cpu - temps CPU max (MIN)
# - nproc - nombre maximal de processus
# - as - adresse limite d'espace (Ko)
# - maxlogins - nombre maximal de connexions pour cet utilisateur
# - maxsyslogins - nombre maximal de connexions sur le système
# - priority - la priorité pour exécuter le processus utilisateur avec
# - locks - nombre maximal de verrous de fichiers que l'utilisateur peut
détenir
# - sigpending - nombre maximum de signaux en attente
# - msgqueue - mémoire maximale utilisée par les files d'attente de messages
POSIX (octets)
# - nice - max nice priorité autorisée à augmenter aux valeurs: [-20, 19]
# - rtprio - priorité maximale en temps réel
# - chroot - change la racine en répertoire (spécifique à Debian)
#
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
#root            hard    core            100000
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4

*               soft    memlock         -1
*               hard    memlock         -1
*               soft    stack           -1
*               hard    stack           -1
*               soft    nofile          16384
*               hard    nofile          16384
*       soft    nproc         16384
*       hard    nproc         16384

# End of file
EOF
```

```
sidus apt-get install nfs-common
mv ${SIDUS}/etc/default/nfs-common ${SIDUS}/etc/default/nfs-common.orig
mv ${SIDUS}/etc/idmapd.conf ${SIDUS}/etc/idmapd.conf.orig
cp /etc/default/nfs-common ${SIDUS}/etc/default/nfs-common
cp /etc/idmapd.conf ${SIDUS}/etc/idmapd.conf
```

Démontage du lien avec le système hôte

```
sidus umount -l /proc
sidus umount -l /sys
sidus umount -l /dev/pts
sidus dpkg-reconfigure dracut
```

Rajout de AUFS dans /etc/updatedb.conf

```
sidus mv /etc/updatedb.conf /etc/updatedb.conf.orig
wget -O ${SIDUS}/etc/updatedb.conf

PRUNE_BIND_MOUNTS="yes"
# PRUNENAMES=".git .bzr .hg .svn"
PRUNEPATHS="/tmp /var/spool /media /.sidus/ro /.sidus/rw"
PRUNEFS="NFS aufs nfs nfs4 rpc_pipefs afs binfmt_misc proc smbfs autofs
iso9660 ncpfs coda devpts ftpfs devfs mfs shfs sysfs cifs lustre tmpfs usbfs
udf fuse.glusterfs fuse.sshfs curlftpfs"
```

Création des utilisateurs et des groupes

- Rajout des groupes système dans /etc/security/group.conf

```
cp ${SIDUS}/etc/security/group.conf ${SIDUS}/etc/security/group.conf.orig
sed -i "s/\#\
End/\*;\*;\*;Al0000-2400;plugdev,fuse,scanner,dialout,video,audio,floppy,cdr
om\n\#\ End/g" ${SIDUS}/etc/security/group.conf
```

- Créer les comptes utilisateurs

les comptes créés sont les 24 lettres grecques, et les UID créés le sont des UID 1001 à 1024 :

Nettoyage du système de fichier

Pour gagner de l'espace on peut supprimr quelques éléments inutiles

```
$ chroot ${SIDUS}
$ export LANG=C
```

```
$ dpkg --get-selections | more
$ apt-get remove --purge cron logrotate  aptitude tasksel tasksel-data
dmidecode laptop-detect
$ rm -f /var/lib/apt/lists/{ftp,http}.* /var/cache/apt/*.bin # au besoin
apt-get update
$ rm -rf /usr/share/{man,doc}/* # pas besoin ;-)
$ rm -rf /usr/share/{locale,zoneinfo}/* # pas besoin :-p
$ rm -rf /usr/lib/gconv/* # encore des locale
$ exit
```

Création de l'image initrd

```
sidus update-initramfs -k /tftpboot/vmlinuz  -u
```

Copier l'image initrd dans `/tftboot/nfs` (par ex. pour une architecture amd64)

```
cp ${SIDUS}/boot/initrd.img-3.2.0-4-amd64 /tftboot/nfs/initrd.img
```

Vérifier que le démarreur comprend bien les composants demandés

```
sidus lsinitrd /boot/initrd.img | egrep '(nfs|aufs)'
```

Préparation d'un initramfs

La seule fonction d'un initramfs est de monter le système de fichier racine. L'initramfs est un ensemble complet de répertoires que l'on peut trouver dans un système de fichiers racine normal. Il est regroupé dans une seule archive cpio et compressé avec l'un des algorithmes de compression.

Au moment du démarrage, le chargeur de démarrage charge le noyau et l'image initramfs dans la mémoire et démarre le noyau. Le noyau vérifie la présence d'un initramfs et, s'il le trouve, le monte sur / et lance /init. Le programme init est typiquement un script shell.

le processus de démarrage est plus long, même significativement plus long, si un initramfs est utilisé.

Construction de l'initramfs

Définition des variables

Définir la version du noyau (optionnel):

```
KERNEL_VERSION=XXXX
```

Définir les binaires à installer dans /bin

```
binfiles="sh cat cp dd killall ls lspci mkdir mknod mount "
binfiles="$binfiles nc umount sed sleep ln rm uname"
binfiles="$binfiles readlink basename bash busybox "
```

Systemd installe udevadm dans / bin. D'autres implémentations d'udev l'ont dans /sbin

```
if [ -x /bin/udevadm ] ; then binfiles="$binfiles udevadm"; fi
```

Définir les binaires à installer dans /sbin

```
sbinfiles="modprobe blkid switch_root"
```

Définir les variables d'environnement

```
DATADIR=/usr/share/mkinitramfs
INITIN=init.in
```

Créer un répertoire de travail temporaire

```
WDIR=$(mktemp -d /tmp/initrd-work.XXXXXXXXXX)
```

Créer une structure de répertoire de base

```
mkdir -p $WDIR/{bin,dev,lib/firmware,run,sbin,sys,proc,usr}
mkdir -p $WDIR/etc/{modprobe.d,udev/rules.d}
touch $WDIR/etc/modprobe.d/modprobe.conf
ln -s lib $WDIR/lib64
ln -s ../bin $WDIR/usr/bin
```

Créer les nœuds de périphérique nécessaires

```
mknod -m 640 $WDIR/dev/console c 5 1
mknod -m 664 $WDIR/dev/null    c 1 3
mknod -m 660 $WDIR/dev/ram0    b 1 0
```

Installer les fichiers de configuration d'udev

```
if [ -f /etc/udev/udev.conf ]; then
  cp /etc/udev/udev.conf $WDIR/etc/udev/udev.conf
fi

for file in $(find /etc/udev/rules.d/ -type f) ; do
  cp $file $WDIR/etc/udev/rules.d
done
```

Installer tout firmware présent

```
cp -a /lib/firmware $WDIR/lib
```

Installer le fichier init

```
install -m0755 $DATADIR/$INITIN $WDIR/init

if [  -n "$KERNEL_VERSION" ] ; then
  if [ -x /bin/kmod ] ; then
    binfiles="$binfiles kmod"
  else
    binfiles="$binfiles lsmod"
    sbinfiles="$sbinfiles insmod"
  fi
fi
```

Installer les binaires de base

Installer les binaires de /bin

```
for f in $binfiles ; do
  if [ -e /bin/$f ]; then d="/bin"; else d="/usr/bin"; fi
  ldd $d/$f | sed "s/\t//" | cut -d " " -f1 >> $unsorted
  cp $d/$f $WDIR/bin
done
```

pour le support de lvm ajouter lvm et dmsetup:

```
if [ -x /sbin/lvm ] ; then sbinfiles="$sbinfiles lvm dmsetup"; fi
```

Installer les binanires de /sbin

```
for f in $sbinfiles ; do
  ldd /sbin/$f | sed "s/\t//" | cut -d " " -f1 >> $unsorted
  cp /sbin/$f $WDIR/sbin
done
```

Ajouter les bibliothèques udevd

si elles ne sont pas dans /sbin

```
if [ -x /lib/udev/udevd ] ; then
  ldd /lib/udev/udevd | sed "s/\t//" | cut -d " " -f1 >> $unsorted
elif [ -x /lib/systemd/systemd-udevd ] ; then
  ldd /lib/systemd/systemd-udevd | sed "s/\t//" | cut -d " " -f1 >>
$unsorted
fi
```

Ajouter des liens symboliques de module si nécessaire

```
if [ -n "$KERNEL_VERSION" ] && [ -x /bin/kmod ] ; then
  ln -s kmod $WDIR/bin/lsmod
  ln -s kmod $WDIR/bin/insmod
fi
```

Ajouter lvm si présent

Ajouter des liens symboliques lvm si nécessaire et copier le fichier `lvm.conf`

```
if  [ -x /sbin/lvm ] ; then
  ln -s lvm $WDIR/sbin/lvchange
  ln -s lvm $WDIR/sbin/lvrename
  ln -s lvm $WDIR/sbin/lvextend
  ln -s lvm $WDIR/sbin/lvcreate
  ln -s lvm $WDIR/sbin/lvdisplay
  ln -s lvm $WDIR/sbin/lvscan

  ln -s lvm $WDIR/sbin/pvchange
  ln -s lvm $WDIR/sbin/pvck
  ln -s lvm $WDIR/sbin/pvcreate
  ln -s lvm $WDIR/sbin/pvdisplay
  ln -s lvm $WDIR/sbin/pvscan

  ln -s lvm $WDIR/sbin/vgchange
  ln -s lvm $WDIR/sbin/vgcreate
  ln -s lvm $WDIR/sbin/vgscan
  ln -s lvm $WDIR/sbin/vgrename
```

```
  ln -s lvm $WDIR/sbin/vgck
  # Conf file(s)
  cp -a /etc/lvm $WDIR/etc
fi
```

Installer les libraries

```
sort $unsorted | uniq | while read library ; do
  if [ "$library" == "linux-vdso.so.1" ] ||
     [ "$library" == "linux-gate.so.1" ]; then
    continue
  fi

  cp $library $WDIR/lib
done

if [ -d /lib/udev ]; then
  cp -a /lib/udev $WDIR/lib
fi
if [ -d /lib/systemd ]; then
  cp -a /lib/systemd $WDIR/lib
fi
```

Installer les modules du noyau si nécessaire

```
if [ -n "$KERNEL_VERSION" ]; then
  find
\
     /lib/modules/$KERNEL_VERSION/kernel/{crypto,fs,lib}
\
     /lib/modules/$KERNEL_VERSION/kernel/drivers/{block,ata,md,firewire}
\
/lib/modules/$KERNEL_VERSION/kernel/drivers/{scsi,message,pcmcia,virtio} \
     /lib/modules/$KERNEL_VERSION/kernel/drivers/usb/{host,storage}
\
     -type f 2> /dev/null | cpio --make-directories -p --quiet $WDIR

  cp /lib/modules/$KERNEL_VERSION/modules.{builtin,order}
\
             $WDIR/lib/modules/$KERNEL_VERSION

  depmod -b $WDIR $KERNEL_VERSION
fi
```

Création de l'initramfs

```

```

```
cd $WDIR
find . | cpio -o -H newc --quiet | gzip -9 ) > /tftpboot/initrd.img
```

Supprimer le répertoire et les fichiers temporaires

```
rm -rf $WDIR $unsorted
```

Configuration finale du serveur

Configuration des exports nfs

Pour permettre l'accès aux fichiers via NFS, configurer le serveur NFS via le fichier `/etc/exports`. Ajouter une ligne comme celle qui suit:

```
/tftpboot   192.168.1.0/24(rw,no_root_squash)
```

Où 192.168.1.0 est le réseau des postes clients.

L'option d'exportation du serveur NFS doit spécifier l'option "no*root*squash", comme expliqué dans mon entrée de blog précédente.

Pour vérifier son intégrité, on peut la monter sur le même ordinateur à l'aide de la commande mount:

```
$ mkdir ~/o755
$ sudo mount 192.168.2.1:/export/root/o755 ~/o755
```

Paramétrage du client lors du boot

Pour amorcer un client sans disque, son noyau doit avoir les options de ligne de commande suivantes:

- **Init=/sbin/init**: Si le programme init est ailleurs, changer simplement le chemin.
- **Root=/dev/nfs**: un alias pour dire au noyau qu'il doit monter son répertoire racine sur nfs
- **Ip**: Cette option de ligne de commande indique au noyau comment obtenir son adresse IP et quelle est l'adresse du serveur NFS.
- **Nfsroot**: indique au noyau de monter ce répertoire en tant que racine.Le% est un alias de l'adresse IP de l'hôte.
- **Vga**: pour pouvoir démarrer X Windows en mode framebuffer (celui-ci correspond à 1024×768 à 16 millions de couleurs).

Editer le fichier de configuration par défaut

```
sudo vi /tftpboot/pxelinux.cfg/default
```

Ajouter ce qui suit:

```
DEFAUT menu.c32
PROMPT 0
MENU TITLE Menu de démarrage PXE
TIMEOUT 50 #Cela signifie 5 secondes apparemment
LABEL buildroot
  MENU LABEL noyau de construction
  KERNEL vmlinuz
  APPEND initrd=initrd.img netboot=nfs nfsroot=192.168.0.50:/tftpboot/nfs
```

Note: Pour certaines distributions, ajouter ce qui suit à la fin de la ligne APPEND:

```
ip=HostIP:ServerIP:RouterIP:Sous-réseau:::none rw
```

Exemple:

```
ip=192.168.0.10:192.168.0.50:192.168.0.1:255.255.255.0:::none rw
```

Démarrage du système de base

A cette étape on doit donc disposer :

- d'un système de base exporter le répertoire /usr/local/linux/base à partir du serveur NFS avec les options rw, no*root*squash.

Tout devrait fonctionner correctement, mais je ne pense pas que vous ayez réussi dès le premier démarrage

Cette méthode standard permet d'amorcer le système de base et d'ajouter des programmes ou un nouveau noyau à votre installation.Vous devez donc sauvegarder les fichiers que vous avez modifiés ainsi que l'image de disque de démarrage.

Après avoir réussi à démarrer le système, vous vous trouvez dans un environnement linux complet: se connecter en tant que root.