LP GSR-Module UE142

Stations sans disques sous Linux

Étude bilan

Pierre Nerzic

IUT de Lannion

1 - Introduction

Un poste dit *client sans disque dur* (diskless) se connecte sur un poste dit *serveur* pour charger un système d'exploitation et travailler avec des logiciels et des fichiers distants. Les calculs (exécution des programmes) sont faits sur le client avec des fichiers situés sur le serveur.

Il faut distinguer cette situation de la connexion d'un terminal X, ! machine ou TS qui est un client également mais sans aucune capacité de calcul, donc les calculs se font sur le serveur.

L'intérêt pour ce cours est la révision de nombreux aspects de l'administration système Linux :

- Configuration et compilation du noyau
- Phases d'initialisation : runlevels et services
- Aspects réseau : installation de serveurs dhcp, tftp et nfs, examen du trafic

- Configuration fine du système : composition et rôles des répertoires Unix
- Administration à distance, dépannage, examen des logs

a) Avantages du fonctionnement diskless

- comme un terminal mais avec une puissance de calcul locale
- économie de disques durs. Il existe maintenant des cartes mères qui contiennent tous les périphériques standards intégrés : processeur, mémoire, carte graphique, carte réseau, carte son, ports usb... pour environ 100€. Il suffit d'acheter un boitier avec alimentation, un clavier, une souris et un écran pour faire un poste diskless totalement silencieux et pas cher.
- pas de répétition des installations dans le cas d’un parc de machines, gestion centralisée des configurations
- pas de sauvegardes à faire de chacun des postes

b) Inconvénients

- nécessité d'avoir des postes matériellement identiques et des besoins identiques. Il serait coûteux de distinguer des configurations.
- charge plus importante du réseau, mais à tempérer par la présence de caches et le type d'activité des utilisateurs (bureautique, calcul ou bdd) – mais de toutes façons, si on désire une mobilité des utilisateurs, il faut monter leurs répertoires par NFS ou SMB...
- très sensible et long à mettre au point : toute erreur dans un fichier de configuration provoque un plantage.

2 - Fonctionnement global

a) Qu'est-ce qu'un système d'exploitation?

Un système est composé de :

- Un noyau qui gère le matériel et la gestion de base des fichiers, des processus... Ce noyau peut être lancé avec des options, ex : hdc=ide-scsi nousb.

- Des systèmes de fichiers (arbres de fichiers montés ensemble), parfois un disque ram initial pour disposer des librairies modules.
- Un ensemble de services : démons et programmes

Démarrer sans disque implique de disposer de tous ces éléments par le réseau.

b) Phases du démarrage

1. démarrage du PC (boot) : demande des infos réseau à un serveur DHCP
2. chargement du noyau : téléchargement sur un serveur TFTP
3. initialisation du matériel et du système : montage d'une racine NFS
4. lancement de init et des services, dont getty pour obtenir une console.

3 - Etape 1 : chargement du noyau client

On va d'abord étudier les deux premières phases du démarrage, ensuite on verra comment compiler et préparer le noyau pour le téléchargement.

a) Aspects réseau

Dans ce cours, on va considérer deux PC en réseau :

- le PC serveur : "serveur.licpro", de numéro IP 192.168.0.1, carte réseau MAC 00:01:01:F7:34:0F
- le PC diskless : "poste.licpro", de numéro IP 192.168.0.2, carte réseau MAC 00:0E:A6:14:8E:82

b) Principe général

Au début, il n'y a rien : le PC diskless n'a pas de disque dur, ne connait pas son numéro IP (ne connait que l'adresse MAC de sa carte réseau), ne sait rien du système d'exploitation qui doit être chargé, ne sait rien des volumes qui seront à monter...

Le BIOS vient d'initialiser la mémoire, la carte mère et les périphériques PCI. En principe, sur une machine habituelle, il va charger en mémoire le MBR du disque de démarrage puis l'exécuter. Mais en mode diskless, il n'y a pas de MBR, donc :

- Le PC publie son n° MAC et attend ses caractéristiques IP de la part du serveur DHCP du réseau.

- Le PC demande ensuite les éléments de base du système à charger : IP du serveur TFTP, nom du fichier à importer.

On passe ensuite au téléchargement du noyau.

c) Chargeurs réseau

Voici les principaux outils qui existent pour faire tout ça :

i) Carte réseau PXE (Preboot eXecution environnement)

C'est un morceau de BIOS placé sur la carte réseau du client connaissant les protocoles DHCP et TFTP. Ce protocole procède en quatre sous-étapes :

- PXE fait une requête DHCP pour obtenir du serveur DHCP son n°IP, le nom d'un programme de boot à charger, par exemple pxelinux.0 et le n°IP du serveur TFTP à contacter (le serveur TFTP n'est pas obligatoirement le serveur DHCP).
- téléchargement de ce programme pxelinux.0 venant du serveur TFTP. Doc : http://syslinux.zytor.com/pxe.php
- ce petit programme cherche (par tftp) un fichier de configuration appelé /tftpboot/pxelinux.cfg/n°IPduclient ou idem.../default. Ce fichier de configuration indique le nom du noyau à charger. Il ressemble beaucoup à lilo.conf.

Doc: http://syslinux.zytor.com/faq.php

- pxelinux.0 charge ensuite le noyau indiqué par le fichier de configuration, il charge aussi son initrd.img, puis le lance avec les options demandées.

On ne peut pas charger directement le noyau car le bios PXE ne sait pas quoi en faire : où le mettre en mémoire, comment le lancer, comment

gérer les options de lancement, comment le relier au disque ram initial (initrd).

ii) Etherboot (www.etherboot.org)

Etherboot est un logiciel qui fonctionne comme précédemment, mais beaucoup plus simplement :

- PXE a obtenu du serveur DHCP son n°IP et le nom du fichier-noyau à charger, par exemple noyau.diskless. Ce fichier n'est pas le noyau pur, c'est un container qui protège le noyau pendant le transport par tftp et qui sait le lancer.
- lancement du fichier-noyau par Etherboot. Le fichier-noyau encapsule le noyau bzImage : comme le noyau est compressé, qu'il doit être lancé avec des options, qu'il peut y avoir un ramdisk, qu'il faut vérifier qu'il n'y a pas eu d'erreur de téléchargement (tftp fonctionne sur UDP), on doit emballer le noyau, le ramdisk et la liste des options dans un fichier spécifique. C'est la famille de commandes mknbi qui fait cela.

Etherboot peut être placé soit dans le BIOS en reprogrammant la carte mère, soit sur une disquette (ses bootblock+premiers secteurs = Etherboot) :

make bin/*carteréseau*.zfd0

Par exemple : make bin/3c905.zfd0.

Si la carte réseau n'est pas connue, on peut essayer soit de modifier l'un des drivers (il suffit souvent de rajouter l'identifiant PCI), soit d'essayer un autre driver. En général, les cartes réseaux ont un fond commun : ce sont les mêmes puces qui resservent avec des améliorations mineures.

iii) Grub

Selon la doc, ça doit marcher ; en réalité, lorsque grub sera enfin compilable et opérationnel avec le support réseau... Sa commande dhcp recherche les caractéristiques du noyau, comme Etherboot puis les commandes root (nd), kernel et boot le chargent et le lancent.

iv) Dans ce TP

On va utiliser la première solution :

- utiliser le bios de la carte mère pour le protocole PXE (Nvidia Boot Agent). Dans le BIOS du client, modifier la config du SouthBridge : activer la carte réseau locale et sa rom, ensuite activer le boot réseau en premier (désactiver tout autre).

  Ensuite, on configurera le serveur pour répondre aux demandes PXE du client.

d) Serveurs à mettre en place sur le poste serveur

D'abord, faire un test de la ROM PXE du client en lançant wireshark avec le filtre « port 67 or port 68 or port 69 ». Vous verrez les requêtes des clients. Le votre doit être dans la liste, repérez son n°MAC. À ce stade, il doit échouer puisqu'il n'y a pas de serveur DHCP.

i) Serveur DHCP

Le serveur DHCP (Dynamic Host Configuration Protocol) permet, entre autres, d'attribuer des adresses IP fixes ou dynamiques à des machines qui démarrent. C'est une évolution du protocole BOOTP. Le protocole DHCP permet aussi de fournir quelques informations aux clients : nom

d'un fichier à télécharger, nom ou n°IP d'un serveur TFTP, racine NFS à monter...

La syntaxe du fichier de configuration permet de nombreuses choses.

Voici un exemple de configuration du serveur : **/etc/dhcpd.conf** :

```
ddns-update-style none;
allow bootp;
allow booting;
deny unknown-clients;
option domain-name "lpgsr";
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
use-host-decl-names on;

subnet 192.168.0.0 netmask 255.255.255.0 {
   option subnet-mask 255.255.255.0;
   option broadcast-address 192.168.0.255;
}

group {
   host serveur {
```

```
      hardware ethernet 00:01:01:F7:34:0F;
      fixed-address 192.168.0.1;
   }
}

group {
   server-name "serveur";
   filename "pxelinux.0";
   option root-path "/diskless";

   # informations pour le boot du client diskless
   host poste {
      hardware ethernet 00:0E:A6:14:8E:82;
      fixed-address 192.168.0.2;
   }
}
```

Ce fichier montre que le poste MAC 00:0E:A6:14:8E:82 se verra attribuer l'adresse IP 192.168.0.2 et le nom "poste.lpgsr". Le fichier qu'il devra demander au serveur tftp "serveur" s'appelle pxelinux.0 et la racine qu'il utilisera sera montée par NFS sur serveur:/diskless.

Sur Ubuntu, il faut éditer /etc/default/dhcp3-server en rajoutant eth0 et activant le service.

Il faut penser à relancer le serveur si on modifie son fichier de configuration :

- **`/etc/init.d/dhcpd restart`** ou **`force-reload`**

Penser aussi à autoriser les accès dhcp par le pare-feu s'il y en a un.

Faire les tests avec wireshark.

ii) Serveur TFTP

Le serveur TFTP (Trivial File Transfer Protocol) permet, comme ftp, de télécharger des fichiers entre un serveur et un client. Seule la commande get est utile pour notre problème. Les clients tftp sont automatiquement placés dans le répertoire **/var/lib/tftpboot**, ainsi que le précise l'option -s ; ils ne peuvent pas télécharger d'autres fichiers que ceux de ce répertoire. En principe pour la sécurité du système, on doit créer un utilisateur spécifique pour les accès tftp et ne donner d'accès à cet utilisateur que sur le répertoire /var/lib/tftpboot.

Il y a au moins deux manières de gérer ce service : xinetd ou autonome.

Si ce serveur est géré par le service **/etc/rc.d/init.d/xinetd**, il est automatiquement lancé lors d'un accès UDP sur le port 69, pourvu qu'il existe un fichier de configuration **/etc/xinetd.d/tftp** :

```
service tftp
{
   disable = no
   flags = REUSE
   port = 69
   socket_type = dgram
   protocol = udp
   user = root
   server = /usr/sbin/in.tftpd
   server_args = -s /var/lib/tftpboot -v -v
   wait = yes
}
```

Penser à relancer le service si on modifie le fichier de configuration : **/etc/init.d/xinetd restart**

Sinon, il faut éditer le fichier /etc/default/tftp-hpa et lancer le service à la main.

Penser aussi à autoriser les accès tftp par le pare-feu.

Tester le serveur en interne par tftp 127.0.0.1, on doit pouvoir récupérer les fichiers de /var/lib/tftpboot (attention, le client a l'impression d'être au niveau de / mais il est bloqué dans /var/lib/tftpboot).

On peut aussi tester le boot et observer etherboot envoyant des requêtes avec cette commande (ou wireshark) :

```
tcpdump -pev
```

On peut imaginer de gérer plusieurs serveurs tftp, un seul serveur dhcp néanmoins qui distribue les clients sur les serveurs tftp.

Installez le paquet syslinux et copiez le fichier /usr/lib/syslinux/pxelinux.0 dans /var/lib/tftpboot.

A ce stade, le client doit planter parce qu'il ne trouve pas la configuration pxe.

iii) Configuration PXE

Créer un répertoire /var/lib/tftpboot/pxelinux.cfg et créer dedans un fichier appelé default contenant ceci (adapter les numéros IP) :

```
DEFAULT Linux
LABEL Linux
KERNEL bzImage
APPEND root=/dev/nfs nfsroot=192.168.0.1:/diskless ip=dhcp
```

Ces options indiquent au noyau de fonctionner en mode réseau. Y rajouter les options locales, telles que noapic nolapic.

A ce stade, le client trouve le fichier de configuration, mais pas bzImage.

4 - Etape 2 : Préparation du noyau client

On doit compiler le noyau linux de manière à ce qu'il soit téléchargeable et qu'il fonctionne uniquement avec NFS (pas de tentative de lecture d'un disque local).

a) Compilation du noyau diskless

Copier les sources du noyau dans un répertoire spécifique, ex: /usr/src/diskless

Modifier le makefile ou le nom du noyau : EXTRAVERSION = -diskless

Partir d'une configuration correcte par exemple la configuration de base d'Ubuntu ou une qui a été testée, lancer make xconfig pour modifier les options suivantes :

- Network Device Support :
  - Ethernet 10 ou 100Mb
    - Mettre Y pour les cartes réseau présentes sur les PC clients.
    - Mettre N pour toutes les autres.
- Networking Options :
  - Mettre Y pour IP: kernel level autoconfiguration (CONFIG_IP_PNP)
  - Mettre Y pour IP: DHCP Support (CONFIG_IP_PNP_DHCP)
- File Systems :
  - Network File Systems :

Mettre Y pour NFS File System Support (CONFIG_NFS_FS)
Mettre Y pour Provide NFSv3 client support (CONFIG_NFS_V3)
Mettre Y pour Root File System on NFS (CONFIG_ROOT_NFS)

On peut également supprimer tous les modules inutiles au client.

Sauver cette configuration .config dans un endroit sûr afin de ne pas tout reprendre à zéro en cas d'erreur de configuration.

Ensuite compiler, mais sans installer le noyau dans /boot ni compiler les modules :

```
make bzImage
```

Il faut copier bzImage dans /var/lib/tftpboot. Cette fois-ci le client parvient à charger le noyau, mais ça se plante un peu plus loin quand le noyau tente de monter sa racine NFS.

b) Installation du noyau pour etherboot

Avec etherboot le transfert est plus simple, il n'y a pas de fichier de configuration. Le client demande directement le noyau au serveur. Mais il faut quand même préparer le noyau d'une certaine manière dans

/tftpboot, cette longue commande appartient au package *mknbi* et correspond au client etherboot ou PXE (attention c'est une seule ligne de commande) :

```
mkelf-linux arch/i386/boot/bzImage
        --output=/var/lib/tftpboot/noyau.diskless
        --append="debug devfs=mount hdc=ide-scsi nousb"
        --ip=dhcp
        --rootdir=/diskless
```

Les options de mkelf-linux indiquent ce que etherboot devra faire avec le noyau : ce noyau obtiendra ses coordonnées par dhcp et sa racine NFS sera montée sur serveur:/diskless. Ne pas oublier de rajouter des options telles que noapic à la chaine --append.

Si cette préparation par mkelf-linux n'est pas faite, etherboot ne reconnaitra pas le noyau et ne le chargera pas.

Eventuellement on peut ensuite compiler les modules, mais il faut les copier à la main dans l’espace client mais ce n'est pas nécessaire si le serveur possède les mêmes, étant donné qu'on va partager /lib (voir la dernière commande).

```
make modules
cp /usr/src/diskless/System.map
/diskless/boot/System.map-2.4.22-diskless
cd /diskless/boot
ln -fs System.map-2.4.22-diskless System.map
rm -fr /diskless/lib/modules/2.4.22-diskless
cp -Rlx /lib/modules/2.4.22-diskless /diskless/lib/modules
```

Il est avantageux de faire un script de mise à jour avec toutes ces commandes.

5 - Etape 3 : Préparation du système client

La dernière partie est la plus longue et la plus difficile : créer l'arborescence du client.

a) Création de l'espace client

Le client diskless fonctionne en utilisant uniquement le serveur NFS : tous les fichiers du système client doivent se trouver sur le serveur NFS mais il faut impérativement les distinguer : /etc/sysconfig/network ou /etc/fstab par exemple ne peuvent pas être les mêmes sur le client et le serveur et il ne faut surtout pas les confondre.

On propose de créer sur le serveur un répertoire **/diskless** contenant l'arborescence du futur client diskless. Par exemple /diskless/var/log du serveur contiendra le répertoire /var/log du client.

Une partie de cette arborescence pourra être composée de liens physiques afin de ne pas dupliquer les fichiers. Cela impose d'avoir tous ces fichiers sur le même volume physique, donc de prévoir de la place pour cela dès l'installation Linux.

Car l'un des défauts de NFS est de ne pas suivre les liens logiques à travers des volumes différents : le serveur NFS fournit à son client le lien tel quel et non pas le fichier qu'il désigne ; le client est donc incapable d'atteindre le fichier s'il n'est pas dans l'espace exporté. Il aurait été par exemple intéressant de lier logiquement /diskless/usr avec /usr. Malheureusement, NFS ne semble pas encore permettre ce genre de liens et les liens physiques ne peuvent être créés qu'a l'intérieur d'un même volume.

D'autre part, il faut se rappeler que le noyau client diskless ne peut monter qu'un seul volume NFS lors du boot, tant qu'il n'a pas lancé les services.

Voici les répertoires que devra obtenir le client, qui sont donc à créer dans /diskless, attention dans ce qui suit, la racine / correspond à ce qui est visible par NFS :

- /bin, /sbin, /lib : contient les logiciels du système : exécutables et librairies. Ne pas oublier que tout exécutable fait appel à une ou plusieurs dll, par exemple /lib/ld-linux.so.

Pour gagner de la place, les contenus de ces répertoires peuvent être composés de liens physiques :

- ✍ **`cp -Rlx /bin /diskless`**
- ✍ **`cp -Rlx /sbin /diskless`**
- ✍ **`cp -Rlx /lib /diskless`**

- /usr : contient les logiciels utilisateur. Ce répertoire est énorme, il est fâcheux de devoir le recopier, il est conseillé de le stocker sur un autre volume qui sera monté également sur /diskless/usr. (peut-on faire un point de montage dans /diskless/usr vers /usr ?)
- /home : si, comme c'est conseillé, ce répertoire est monté sur un autre volume, on peut faire un second montage sur /diskless/home.
- /etc : ce répertoire devra contenir les fichiers spécifiques au client. Ces fichiers devront être adaptés, voir plus loin.

  ✍ **`cp -Rx /etc /diskless`**

- /dev : ce répertoire est vide car les périphériques sont montés par devfs. Quand devfs ne fonctionne pas ou pas complétement, il faut créer manuellement quelques périphériques : console, null, tty. Voici les commandes :

  - **`mknod /diskless/dev/console c 5 1`**

  - **`mknod /diskless/dev/tty0 c 4 0`**

  - **`mknod /diskless/dev/null c 1 3`**

Explications : les éléments de /dev sont des pseudo-fichiers : ce sont des éléments qui représentent les périphériques du système. Ecrire dans /dev/console à pour effet d'écrire sur l'écran, lire /dev/tty1 interroge le clavier de la première page (alt-F1). Les périphériques sont donc visibles sous forme de fichiers et le lien avec le système se fait par les deux numéros : majeur et mineur, par exemple 5 1 pour accéder à la console. Chaque périphérique a ses numéros. Pour les connaitre, il faut lister /dev avec l'option –l. Cela fait apparaitre le type du périphérique : caractère ou bloc. Certains périphériques sont des flux de caractères (mode c), d'autres sont des ensembles de blocs (mode b) tels que les disques durs.

- /var, /tmp : ces répertoires contiennent des fichiers temporaires. Certains fichiers de /var doivent être créés avant le boot (utmp...). Ne pas copier /var/lib/tftpboot.

  ✍ **`cp -Rx /var /diskless`**

  ✍ **`mkdir /diskless/tmp`**

- /mnt, /net, /opt... : répertoires divers
- /boot : contient quelques fichiers liés au noyau : symboles...

NB : dans le cas de plusieurs clients diskless, il devient important qu'ils puissent distinguer certains répertoires, comme /etc, /var et /tmp.

- /var/lib/tftpboot : ce répertoire est la racine du serveur tftp, voir plus loin. On y met le noyau à télécharger.

b) Initialisation du noyau

On se trouve maintenant après le chargement du noyau en mémoire par pxe, etherboot ou grub.

Le noyau commence par initialiser tout le matériel : processeur, bus PCI, ports USB... Le listing apparait très vite.

Etherboot ou PXE lui a fourni les options -ip=dhcp -root=/dev/nfs qui lui indiquent de rechercher le serveur NFS par DHCP et de monter / sur le répertoire indiqué du serveur NFS, ex: serveur:/diskless (échecs potentiels : serveur NFS inactif, pare-feu bloquant, racine pas exportée).

Ensuite, on arrive au point critique : le lancement d'init et des services. C'est une étape délicate car il se peut que les montages NFS réalisés ne fonctionnent pas. Si c'est le cas, le message « kernel panic, no init found » apparait.

Par exemple, si on a monté et exporté /usr sur un autre volume, le système ne trouvera pas les fichiers /usr/bin, /usr/lib... tant que ce volume ne sera pas monté.

Pour le montage de volumes NFS deux fichiers sont concernés :

serveur:**/etc/exports** : indique quels volumes doivent être offerts aux clients

```
/diskless 192.168.0.0/255.255.255.0(rw,no_root_squash,sync)
/home     192.168.0.0/255.255.255.0(rw,no_root_squash,sync)
```

```
/usr      192.168.0.0/255.255.255.0(rw,no_root_squash,sync)
```

poste:/etc/fstab == serveur:**/diskless/etc/fstab**

```
192.168.0.1:/diskless   /      nfs rsize=8192,wsize=8192,timeo=14,intr
192.168.0.1:/home       /home  nfs rsize=8192,wsize=8192,timeo=14,intr
192.168.0.1:/usr        /usr   nfs rsize=8192,wsize=8192,timeo=14,intr
```

root squash or not root squash ? that is the question...

Des essais montrent qu'il vaut mieux ne monter aucun des /lib, bin, /sbin, /root et /tmp. Le passage d'un volume plus ou moins local (sur la racine NFS) à un volume entièrement NFS en cours de démarrage du noyau pose problème. En fait, c'est un peu regrettable, mais il faudrait que le noyau soit capable de monter plusieurs volumes NFS à différents endroits et pas seulement la racine avant de lancer init, car les nombreux logiciels qui suivent ont besoin de pratiquement toute l'arborescence Unix, on est donc obligé de tout recopier dans /diskless.

c) Init et les runlevels

On rappelle le déroulement dont les étapes sont fixées par des scripts de **/etc** :

ATTENTION : le client diskless travaille avec SERVEUR:/diskless/etc et non pas /etc directement ; ne surtout pas confondre ces deux répertoires !

1. lancement d'init (qui a besoin de plusieurs DLL, p. ex. /lib/ld.linux.so, /lib/i686/libc.so...)
2. init consulte **/etc/event.d** qui indique quels scripts lancer dans quelles circonstance : boot, arrêt...
3. le boot entraine le lancement des scripts **/etc/rcS.d/**
4. ensuite, on passe dans le runlevel indiqué dans inittab, p. ex: 2 et on exécute des stop sur les services **/etc/rc.d/rcN.d/K*** puis des start sur les **/etc/rc.d/rcN.d/S***. Ces K* et S* sont des liens logiques sur les scripts de **/etc/rc.d/init.d**.
5. enfin on donne la main à des mingetty sur les consoles tty...

d) Quelques services à configurer spécialement

On rappelle quelques services à connaitre :

- /etc/rc.d/init.d/network démarre les aspects réseau et interface eth0

Il faut modifier le fichier /diskless/etc/sysconfig/network et mettre le bon nom de la machine. Dans /diskless/etc/sysconfig/network-scripts/ifcfg-eth0, mettre le bon n°IP (on pourrait écrire un script qui récupère ces informations du noyau sans avoir besoin de ces scripts de configuration).

- /etc/rc.d/init.d/portmap démarre le service RPC nécessaire à NFS, exige que network soit actif
- /etc/rc.d/init.d/netfs monte les volumes NFS mentionnés dans **/etc/fstab**, exige portmap actif
- /etc/rc.d/init.d/xinetd démarre l'observateur des ports internet

Certains services peuvent/doivent être arrêtés ou reconfigurés :

- xfs (serveur de fontes X11), cron, random…
- les serveurs dhcp, tftp sur le client.

On propose d’utiliser un runlevel spécifique pour les clients, p. ex : 4 dans lequel on a limité les services lancés au strict minimum.

La mise au point de tout ceci n’est pas facile et demande beaucoup de temps.

6 - Résumé pour la mise en oeuvre

a) Paquetages à installer

- éventuellement PXE ou etherboot compilé pour la bonne carte réseau.
- mknbi installé, ce logiciel accompagne aussi etherboot.
- un serveur dhcp complet : dhcp3-server
- un serveur tftp : tftpd-hpa et tftp-hpa
- un serveur nfs : nfs-kernel-server
- des outils réseau : wireshark

b) Etapes

1.configurer le bios du client en mode boot réseau

test : le client émet des demandes sur le réseau mais ne reçoit pas (encore) de réponse

2. configurer le serveur DHCP pour ne répondre qu'à ce client, envoi du n°IP, du serveur TFTP et du nom du noyau.

test : le client reçoit un n°IP puis s'adresse au serveur TFTP, mais sans succès

3. configurer le serveur TFTP

test : le client s'adresse au serveur TFTP mais ne parvient pas à télécharger un noyau.

4. créer le noyau diskless et le placer sur le serveur TFTP

test : le client télécharge le noyau, commence à démarrer mais échoue sur 'no init found'

5. configurer le serveur NFS (très long)

test : le client démarre et fait apparaître une offre de connexion.

Il est demandé d'arriver à se connecter en mode texte sur le poste client.