

Création d'un serveur de «Terminaux X» avec Gentoo

Rédigé par

Steeve Maltais

steevem@levinux.org

Publié par

Jacques Daignault

jacques@levinux.org

<http://publications.levinux.org>

Historique des versions		
1.1	24 juillet 2006	JD
Modifications mineures aux instructions, ajout de meta-données et révision de la mise en page des "programlisting"		
1.00	1er Mars 2004	SM
Document réalisé dans le cadre de la maîtrise en technologie éducative de l'auteur (stage à Levinux).		
Mention légale		
Licence : Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation libre GNU (GNU Free Documentation License), version 1.1 ou toute version ultérieure publiée par la Free Software Fondation; sans Sections Invariables; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est incluse dans la section intitulée « Licence de Documentation Libre GNU » (en hyperlien dans le pied-de-page du site publications.LEVINUX.org).		
Copyright : ©2004 ©2006; ©Steeve Maltais ©Levinux		

Résumé

Ce document contient une procédure d'installation possible pour créer un serveur pour «Terminaux X». Les «terminaux X» sont des ordinateurs branchés en réseau, de classe Pentium 1 ou supérieur. Ils ne sont pas autonomes, au sens de leur capacité à fonctionner par eux-mêmes car ils ne contiennent pas de disque rigide. Par contre, ils deviennent autonomes dès qu'ils sont branchés en réseau avec un serveur agissant à titre de «fournisseur de système d'exploitation» pour ces même terminaux. Cela revient donc à dire qu'un laboratoire complet peut fonctionner avec un serveur puissant, de classe Pentium 4, et alimenter un laboratoire d'une douzaine d'appareils sans problèmes et ce, avec tout ce que cela comporte : Interface graphique (KDE ou Gnome), navigateur web, courriel, interpréteur de commandes, édition graphique,

traitement de texte, etc. Enfin, tout ce qui existe comme logiciels sous Linux! Le but de ce document est de vous amener à construire vous même ce serveur de Termiaux X. Prenez note que vous aurez à rééditer à plusieurs reprises les mêmes fichiers et ceci est tout à fait volontaire de notre part : cette procédure vous permettra de tester ce qui fonctionne bien ou moins bien et ce, au fur et à mesure que nous avancerons. Ce document représente le fruit d'un travail de collaboration étroit entre plusieurs partenaires de Lélinux , laboratoire d'éducation à la virtualité situé au campus de l'UQAR à Lévis. Ce document sous-entend que vous avous déjà un système Gentoo fonctionnel. Pour plus d'informations à ce sujet consultez les autres documents disponibles sur le site de Lélinux. N'hésitez pas à vous référer au site de Linux Gentoo si nécessaire: [Documentation officielle de Gentoo](#)

Table des matières

1. Architecture réseau	1
1.1. Représentation visuelle	1
1.2. Description du réseau	2
2. Noyau OpenMosix	3
2.1. Pré-Installation	3
2.2. Emerger les sources	3
2.3. Configuration du noyau	4
2.4. Compilation du noyau	4
3. Outils OpenMosix	6
3.1. Installation	6
4. Diskless	8
4.1. Paramètres du noyau	8
4.2. Compilation du noyau : diskless	9
4.3. Compilation du noyau : terminaux	9
4.4. Édition des fichiers nécessaires	10
4.5. Activation du DHCP	13
4.6. tFTP et démarrage via PXE : Configuration	13
4.6.1. Création de l'environnement pour PXE	14
4.7. Serveur NFS : Configuration	14
5. LTSP 4	17
5.1. Monter et configurer LTSP	17
5.2. Modification finale au fichier exports	19
5.3. Configuration du DHCP	19
5.4. Édition en vrac de quelques fichiers	22
5.5. Voilà!	22

Liste des illustrations

1.1. Architecture réseau	1
--------------------------------	---

Chapitre 1. Architecture réseau

1.1. Représentation visuelle

Voici la représentation visuelle du réseau que vous aurez à construire une fois que votre serveur «rouge» sera fonctionnel. Ce serveur contient 3 cartes réseau (eth0, eth1, eth2) ayant chacune une utilité particulière.

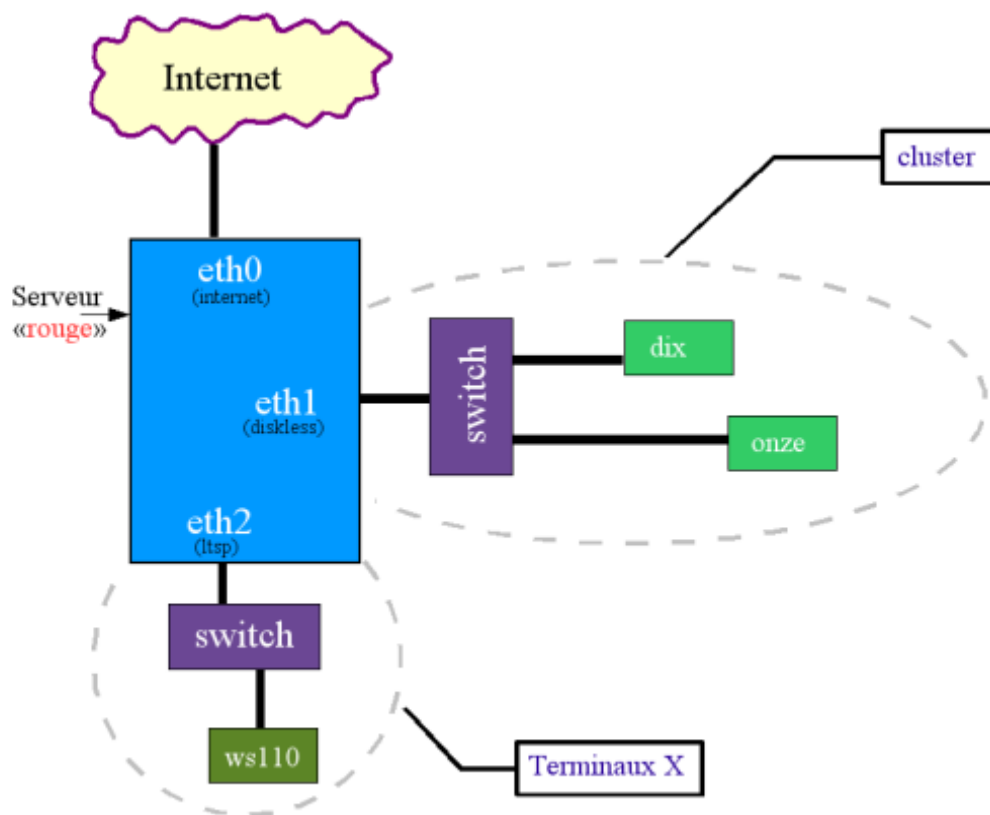


Figure 1.1. Architecture réseau

1.2. Description du réseau

Voici à quoi ressemble la configuration des cartes réseau de votre réseau. Ces éléments de configuration seront écrits aux emplacements appropriés plus loin dans ce document. Cette section n'est qu'à titre informatif...

Serveur

```
hostname = rouge
eth0 : 192.168.0.50 // Relié à Internet
eth1 : 192.168.50.1 // Relié au cluster
eth2 : 192.168.51.1 // Relié aux terminaux X
gateway = 192.168.0.1 // Correspond à votre serveur et votre adresse gateway
                    sur votre réseau (changer au besoin)
dns = 192.168.0.14 // Correspond à votre serveur et votre adresse dns sur votre
                    réseau (changer au besoin)
```

noeud 1

```
host name = dix
mac = 00:04:75:F4:E6:BC //Mac adresse de votre carte réseau. Adapter selon
                    votre matériel!
ip = 192.168.50.10
```

noeud 2

```
hostname = onze
mac = 00:04:75:F4:EC:07
ip = 192.168.50.11
```

terminal

```
hostname = ws100
mac = 00:0A:E6:0D:00:93
ip = 192.168.51.110
```

Chapitre 2. Noyau OpenMosix

À cette étape nous allons configurer votre serveur pour le rendre capable de supporter ce que nous appelons en informatique le "clustering", soit le "groupage". Pour cela nous utiliserons la fonctionnalité **OpenMosix**¹. Cette fonctionnalité ne nécessite aucun logiciel supplémentaire car elle est fait déjà partie intégrante de votre noyau Linux. Il ne s'agit que de bien configurer ce dernier pour qu'il vous offre ces fonctionnalités! Le clustering est une façon de transformer un groupe d'ordinateur ordinaires en réseau en un super-ordinateur pouvant rouler plus efficacement toute la gamme de vos logiciels linux favoris. En anglais c'est ce qu'on appelle aussi du "load balancing", ce qui se traduirait en français par "équilibrage de charge".

2.1. Pré-Installation

Commençons par installer GCC, le compilateur C et C++ de Linux (entre autres langages)...

```
#emerge -k gcc
```

Note

La version proposée sera la version 3.3.2-r5 ou supérieure. Il serait préférable d'installer la version 2.95. Pour cela il faut ... (Steeve : consigne à ajouter ici!!!)

```
#emerge -k gcc-config
#gcc-config -l // Pour voir les versions de GCC installées
```

Cela devrait vous donner quelque chose en sortie comme ceci :

```
#[1] i686-pc-linux-gnu-3.3.2
```

[1] représentant la version installée sur votre système.

2.2. Emerger les sources

```
#emerge -k openmosix-sources
#cd /usr/src/
#rm -rf linux
#ln -s linux-2.4.22-openmosixlinux // Le nom du dossier peut changer.
                                     // Adapter à la version en cours,
                                     // soit linux-2.x.xx-openmosixlinux
#cd /usr/src/linux
#cp .config .config_working // Conserver l'ancienne configuration du
```

¹ <http://openmosix.sourceforge.net/>

```
kernel
```

```
#make menuconfig
```

2.3. Configuration du noyau

Nous allons maintenant passer aux choix de modules à compiler (ajouter) dans le noyau.

```
OpenMosix -->
```

```
 [*] openMosix process migration support
 [ ] Support clusters with a complex network topology
 [*] Stricter security on openMosix ports
 (1) Level of process-identity disclosure (0-3)
 [*] openMosix File-System
 [ ] Poll/Select exceptions on pipes
 [ ] Disable OOM Killer
 [ ] Load Limit
```

```
Code maturity level options --->
```

```
 [*] Prompt for development and/or incomplete code/drivers
```

```
Networking options --->
```

```
<*> Packet socket
 [ ] Packet socket: mmapped IO
 < > Netlink device emulation
 [ ] Network packet filtering (replaces ipchains)
 [*] Socket Filtering
 <*> Unix domain sockets
 [*] TCP/IP networking
 [*] IP: multicasting
```

```
File systems --->
```

```
 [*] /proc file system support
 [*] /dev file system support (EXPERIMENTAL)
 [*] Automatically mount at boot
```

2.4. Compilation du noyau

Finalement, il faut compiler le noyau. Ceci implique la création des dépendances, création de l'image pour le répertoire boot/, la création des modules, leur installation et finalement copie du fichier bzImage.

```
# make dep
# make clean bzImage modules modules_install
```

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage
```

Chapitre 3. Outils OpenMosix

Pour exploiter les fonctionnalités OpenMosix de votre noyau, il faut installer divers outils. Ces petits logiciels vous permettront de tirer certaines informations de vos noeuds présents sur le réseau. Un noeud est tout simplement un ordinateur présent sur le réseau, recevant des tâches à exécuter du serveur principal.

3.1. Installation

Procédons maintenant à l'installation en tant que telle...

```
#emerge -k openmosix-user //Installer les outils OpenMosix
#mkdir mfs //Créer le point de montage /mfs dans le répertoire racine,
           c'est à dire " / "
```

Modification du fichier /etc/fstab...

```
#nano /etc/fstab
```

Ajoutez la ligne suivante :

```
none          /mfs          mfs           noauto,dfsa=1    0 0
```

Quittez nano en enregistrant les changements et ...

1. Démarrez OpenMosix et
2. Écrivez la commande pour qu'il démarre automatiquement à l'ouverture du système

en utilisant le code suivant :

```
#/etc/init.d/openmosix start
#rc-update add openmosix default
```

Important

Le paquetage OpenMosix-user contient quelques outils bien pratiques. Pour n'en nommer que quelques uns (à exécuter en ligne de commande) :

mosmon : moniteur OpenMosix. Affiche l'état de vos noeuds OpenMosix.

mtop : version améliorée de top qui affiche sur quel noeud les processus tournent.

m�� : version améliorée de ps qui affiche les numéros de noeuds.

mosctl whois : considérant que mosmon n'affiche que les numéros de noeuds, cette commande est très utile car elle affiche aussi l'adresse ip et le nom d'hôte du noeud.

omdiscd eth1 : OpenMosix sur l'interface eth1

Il y a aussi l'application *openmosixview*¹ qui permet d'afficher un aperçu de votre cluster et des processus migrés d'un noeud à l'autre. Vous pouvez l'installer comme ceci...

```
#emerge -k openmosixview
```

¹ <http://openmosix.sf.net>

Chapitre 4. Diskless

Comme pour OpenMosix au chapitre 2, il faut configurer le noyau (et le recompiler) en vue de pouvoir monter une partition racine sur le réseau. Contrairement à un poste de travail autonome où la partition racine serait montée sur le disque rigide de l'ordinateur en question.

Qu'est-ce qu'une machine diskless? (Source : Site de Gentoo)

«Une machine diskless est un PC dépourvu des périphériques de démarrage habituels, c-à-d. sans disque dur ni lecteur de CDRom ou de disquette. Un PC diskless démarre sur le réseau et a besoin d'un serveur qui va lui fournir de l'espace disque comme un disque dur local. Nous appellerons le serveur le maître et les machines sans disques des esclaves. Un esclave a besoin d'une carte réseau qui supporte le démarrage via PXE. La plupart des cartes modernes les font, les adaptateurs réseau intégrés sur les cartes-mères aussi.»

4.1. Paramètres du noyau

```
#cd /usr/src
#cp .config .config_working //sauvegarder une copie de la configuration du
                             noyau maître
#make menuconfig           //C'est ici qu'on modifie la configuration du
                             noyau maître
```

Assurez-vous que votre noyau possède ces options :

```
Code maturity level options --->
  [*] Prompt for development and/or incomplete code/drivers

Networking options --->
  <*> Packet socket
  [ ] Packet socket: mmaped IO
  < > Netlink device emulation
  [ ] Network packet filtering (replaces ipchains)
  [*] Socket Filtering
  <*> Unix domain sockets
  [*] TCP/IP networking
  [*] IP: multicasting

File systems --->
  [*] /proc file system support
  [*] /dev file system support (EXPERIMENTAL)
```

```
[*] Automatically mount at boot
Network File Systems --->
  <*> NFS server support
  [*] Provide NFSv3 server support
```

4.2. Compilation du noyau : diskless

Finalement, il faut compiler le noyau. Ceci implique la création des dépendances, création de l'image pour le répertoire boot/, la création des modules, leur installation et finalement copie du fichier bzImage pour remplacer l'ancien.

```
# make dep
# make clean bzImage modules modules_install
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage
# cp /usr/src/linux/.config /usr/src/linux/.config_master // Sauvegarde d'une
                                                                copie de la
                                                                configuration
                                                                du noyau maître)
```

Nous allons maintenant procéder à la compilation d'un nouveau noyau, noyau qui sera nécessaire à vos terminaux X...

4.3. Compilation du noyau : terminaux

Vous aurez besoin de nouveaux noyaux pour vos terminaux X. En fait, lorsque ceux-ci démarreront, ils utiliseront ce noyau spécialement conçu pour eux.

Avertissement

Assurez-vous que vous sélectionnez les options en tant que composantes internes au noyau et non en tant que modules.

```
Code maturity level options --->
  [*] Prompt for development and/or incomplete code/drivers

Networking options --->
  <*> Packet socket
  [ ] Packet socket: mmaped IO
  < > Netlink device emulation
  [ ] Network packet filtering (replaces ipchains)
  [ ] Socket Filtering
  <*> Unix domain sockets
```

```
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: kernel level autoconfiguration
[*] IP: DHCP support (NEW)
```

File systems --->

```
[*] /proc file system support
[*] /dev file system support (EXPERIMENTAL)
[*] Automatically mount at boot
```

Network File Systems --->

Ajouter toutes les options requises pour les cartes ethernet de vos noeuds esclaves

Procédons maintenant à la compilation :

```
#cd /usr/linux
#make clean dep bzImage //Compilation du noyau esclave
#mkdir /diskless
#cp /usr/src/linux/arch/i386/boot/bzImage /diskless //Copie du noyau esclave
```

4.4. Édition des fichiers nécessaires

Les systèmes de fichiers des noeuds maître et esclaves oeuvent subir de nombreuses adaptations. Concentrons-nous d'abord sur les fichiers de configuration et les points de montage. Nous avons besoin d'un répertoire sous /diskless pour le premier noeud esclave . Chaque esclave a besoin de son propre système de fichiers racine (le "root") parce que certains fichiers ne peuvent pas être communs à plusieurs machines sans causer de graves problèmes. Peu importe les noms des sous-répertoires vous pouvez, par exemple, les nommer d'après les adresses IP car elles sont uniques et explicites. Dans cet exemple, l'adresse IP du premier noeud esclave est 192.168.50.10 : vous devez faire cette opération pour chaque noeud esclave. Voir le code ci-bas.

Qu'est-ce qu'un noeud esclave?

C'est un ordinateur qui reçoit des tâches à effectuer par «l'ordinateur maître» !

```
# mkdir /diskless/192.168.50.10
# cp -r /etc /diskless/192.168.50.10/etc
# mkdir /diskless/192.168.50.10/home
# mkdir /diskless/192.168.50.10/dev
# mkdir /diskless/192.168.50.10/proc
# mkdir /diskless/192.168.50.10/tmp
# mkdir /diskless/192.168.50.10/mnt
# mkdir /diskless/192.168.50.10/mnt/.initd
# mkdir /diskless/192.168.50.10/root
```

```
# mkdir /diskless/192.168.50.10/var
# mkdir /diskless/192.168.50.10/var/empty
# mkdir /diskless/192.168.50.10/var/lock
# mkdir /diskless/192.168.50.10/var/log
# mkdir /diskless/192.168.50.10/var/run
# mkdir /diskless/192.168.50.10/mfs
```

Important

Vous devez recommencer cette procédure pour chaque station (noeud esclave) que vous possédez. Vous pouvez incrémenter de 1 pour chaque nouvelle station, comme par exemple : 192.168.50.10 pour la première, 192.168.50.11 pour la deuxième, 192.168.50.12 pour la troisième et ainsi de suite...

Vous devez ensuite modifier le fichier «net» situé dans le répertoire /etc/net pour forcer la carte réseau des postes esclaves à obtenir une adresse ip par dhcp. Encore une fois, cette procédure est nécessaire pour CHAQUE noeud esclave.

```
#nano /diskless/192.168.50.1/etc/net
```

Gardez seulement les lignes suivantes :

```
iface_eth0="dhcp"
gateway="eth0/192.168.50.1" //Changez le ip pour celui de votre passerelle
```

Quittez le fichier et enregistrez les modifications. Nous allons maintenant procéder à l'installation du serveur DHCP. Le serveur DHCP procure une adresse IP de façon automatique à tous les ordinateurs qui se brancheront sur votre réseau.

```
#emerge -k dhcp
```

Modifions maintenant le fichier de configuration du serveur DHCP :

```
#cp /etc/dhcp/dhcp.conf.sample /etc/dhcp/dhcp.conf
#nano /etc/dhcp/dhcp.conf
```

Votre fichier dhcp.conf, version finale, devrait ressembler à ceci :

```
ddns-update-style none;

option space PXE;
option PXE.mtftp-ip                code 1 = ip-address;
option PXE.mtftp-cport              code 2 = unsigned integer 16;
option PXE.mtftp-sport              code 3 = unsigned integer 16;
option PXE.mtftp-tmout              code 4 = unsigned integer 8;
option PXE.mtftp-delay              code 5 = unsigned integer 8;
option PXE.discovery-control        code 6 = unsigned integer 8;
```

```
option PXE.discovery-mcast-addr code 7 = ip-address;

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

subnet 192.168.50.0 netmask 255.255.255.0
{
    class "pxeclients"
    {
        match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
        option vendor-class-identifier "PXEClient";
        vendor-option-space PXE;
        option PXE.mtftp-ip 0.0.0.0;
        filename "pxelinux.0";
        next-server 192.168.50.1;                //adresse du serveur
    }

    pool
    {
        max-lease-time 86400;
        default-lease-time 86400;
        deny unknown clients;
    }

    host dix
    {
        hardware ethernet          00:04:75:F4:E6:BC; //adresse mac du client
        fixed-address                192.168.50.10;
        server-name                  "rouge";           //nom du serveur
        option routers               192.168.50.1;     //adresse du serveur
        option domain-name-servers  192.168.0.14;     //adresse du dns
        option host-name              "dix";           //nom du client
    }

    host onze
    {
        hardware ethernet          00:04:75:F4:EC:07;
        fixed-address                192.168.50.11;
        server-name                  "rouge";
        option routers               192.168.50.1;
    }
}
```

```
option domain-name-servers 192.168.0.14;
option host-name            "onze";
}
}

subnet 192.168.51.0 netmask 255.255.255.0
{
    option broadcast-address 192.168.51.255;
}
}
```

4.5. Activation du DHCP

Il faut maintenant spécifier sur quel interface réseau le serveur DHCP devra s'exécuter.

```
#nano /etc/conf.d/dhcp //Exemple de /etc/conf.d/dhcp

IFACE="eth1 eth2"
DHCPD_OPTS=""
```

Quittez nano en sauvegardant les changements.

Démarrez le serveur DHCP et ajoutez-le à la procédure de démarrage pour qu'il s'exécute automatiquement.

```
#/etc/init.d/dhcp start

#rc-update add dhcp default
```

Important

N'oubliez pas de redémarrer le serveur dhcp à chaque modification du fichier dhcpd.conf

```
#/etc/init.d/dhcpd restart
```

4.6. tFTP et démarrage via PXE : Configuration

Le serveur tFTP (Trivial FTP) est utilisé pour envoyer le noyau et le initrd au terminal, via le réseau.

Commençons par installer les services nécessaires :

```
#emerge -k syslinux //Installer syslinux

#emerge tftp-hpa //Installer le serveur tFTP
```

```
#cp /usr/lib/syslinux/pxelinux.0 /diskless //installer le démarreur à distance  
  
#mkdir /diskless/pxelinux.cfg
```

4.6.1. Création de l'environnement pour PXE

Pour chaque appareil de votre réseau (terminal X) vous devez prendre son adresse ip, la convertir en hexadécimal et créer un fichier portant cette adresse convertie comme nom. Comme ceci :

```
#nano /diskless/pxelinux.cfg/C0A8320A //C0A8320A en hexadécimal = 192.168.50.10  
en décimal)
```

Attention, il faut modifier l'adresse IP pour une adresse en format hexadécimal. Lorsque vous la réécrivez, n'incluez pas les " ." tel que vous êtes habitués de la faire comme en écrivant, par exemple, l'adresse 192.168.0.1

Voici un fichier contenant un exemple de contenu pour pxelinux.cfg/C0A8320A

```
DEFAULT /diskless/bzImage  
APPEND ip=dhcp root=/dev/nfs nfsroot=192.168.1.1:/diskless/192.168.1.10
```

Il faut maintenant configurer votre serveur tFTPD pour qu'il démarre avec les bons paramètres. Modifiez pour cela le fichier dans /etc/conf.d/in.tftpd :

```
INTFTPD_PATH="/diskless"  
INTFTPD_OPTS="-l -v -s ${INTFTPD_PATH}"
```

En dernier lieu, il faut démarrer le service et l'ajouter au démarrage du noeud maître (votre serveur) :

```
#/etc/init.d/in.tftpd start  
# rc-update add in.tftpd default
```

4.7. Serveur NFS : Configuration

NFS (Network File System) permet de rendre disponible le système de fichiers aux terminaux. Un "Network File System" (Système de fichiers réseau) est un protocole réseau qui supporte le partage de fichiers, comme par exemple le protocole SMB pour Windows.

Commençons par installer le paquetage requis :

```
#emerge nfs-utils
```

Configurez maintenant le fichier de démarrage du serveur NFS

```
#nano /etc/init.d/nfs
```

Modifiez-le fichier pour qu'il prenne les valeurs suivantes :

```
# Number of servers to be started up by default
RPCNFSDCOUNT=20

#Options to pass to rpc.mountd
RPCMOUNTDOPTS=" "
```

Sortez de nano et enregistrez les changements. Démarrez le serveur NFS et ajoutez-le au processus de démarrage du noeud maître (votre serveur) :

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

#rc-update add nfs default
```

Par la suite, il faut modifier le fichier "exports", fichier contenant tous les principes partagés par NFS.

```
#nano /etc/exports
```

Important

Dans ce fichier, il faut ajouter une ligne pour chacun des noeuds esclave présents sur votre réseau!
Les quatres lignes à ajouter pour chaque terminaux sont celles commençant par /diskless et /var/log

```
# /etc/exports: NFS file systems being exported.  See exports(5).
/diskless/192.168.50.10    192.168.50.10(rw,no_root_squash,no_all_squash, sync)
/diskless/192.168.50.11    192.168.50.11(rw,no_root_squash,no_all_squash, sync)
/opt      192.168.50.0/24(ro,no_root_squash,no_all_squash, sync)
/usr      192.168.50.0/24(ro,no_root_squash,no_all_squash, sync)
/home     192.168.50.0/24(rw,no_root_squash,no_all_squash, sync)
/var/log  192.168.50.10(rw,no_root_squash,no_all_squash, sync)
/var/log  192.168.50.11(rw,no_root_squash,no_all_squash, sync)
```

Par la suite, il faut modifier le fichier "fstab" spécifique à chaque noeud, comme ceci :

```
#nano /diskless/192.168.50.10/etc/fstab
```

Le fichier fstab contenu dans /diskless/192.168.50.10/etc/fstab du noeud esclave que vous configurez devrait ressembler à ceci :

```
192.168.50.1:/diskless/192.168.50.10 /    nfs    hard,intr,rw,nolock,rsize=8192,\
wsize=8192    0 0
192.168.50.1:/opt /opt  nfs    hard,intr,ro,nolock,rsize=8192,\
wsize=8192    0 0
```

```
192.168.50.1:/usr          /usr  nfs   hard,intr,ro,nolock,rsize=8192,\
wsize=8192  0 0
192.168.50.1:/home       /home nfs   hard,intr,rw,nolock,rsize=8192,\
wsize=8192  0 0
192.168.50.1:/var/log    /var/log  nfs   hard,intr,rw                    0 0

# NOTE: The next line is critical for boot!
none      /proc      proc      defaults      0 0
none      /mfs       mfs       noauto,dfsa=1 0 0
none      /dev/shm   tmpfs     defaults      0 0
```

Passons maintenant à la création de certains dossiers nécessaires nécessaires à chaque noeud esclave. Encore une fois, vous devez créer ces dossiers pour chaque noeud esclave de votre réseau et ce, autant que vous en avez. N'oubliez pas de modifier l'adresse ip du noeud esclave en question à chaque fois que vous créez ses dossiers...

```
# cp -r /bin /diskless/192.168.50.10/bin
# cp -r /sbin /diskless/192.168.50.10/sbin
# cp -r /lib /diskless/192.168.50.10/lib
```

Ensuite, pour chaque noeud esclave (selon l'adresse ip correspondante) faites :

```
diskless/192.168.50.10/fastboot
# echo "touch /fastboot" >> /diskless/192.168.50.10/etc/conf.d/local.start
```

Chapitre 5. LTSP 4

LTSP est une application qui permet de brancher plusieurs ordinateurs de type "Terminaux" (Aussi appelés "Terminaux X") à un serveur Linux (le noeud maître). Les applications s'exécutent normalement sur le serveur. Ce dernier reçoit les requêtes des terminaux et retourne les réponses via leur interface graphique. C'est pourquoi un terminal peut être un simple Pentium 1 et se comporter comme si c'était un Pentium 4, dans le cas où votre serveur en serait un...

Passons à l'installation de ltsp en commençant par installer GDM (Gnome Display Manager) et XFS (X Font System) :

```
emerge -k gdm //installer gdm
emerge -k xfs //installer xfs
```

Une fois la compilation et l'installation terminée, ajoutons ces deux services pour qu'il s'exécutent automatiquement au démarrage du serveur :

```
#rc-update add gdm default
#rc-update add xfs default
```

Important

Concernant l'installation du package LTSP, il faut le télécharger directement du site de LTSP. Vous devez télécharger le fichier *ltsp-4.iso* (ou la version courante au moment où vous lisez ces lignes). Voir pour cela le site suivant :

www.ltsp.org¹

Note

Vous devez sauvegarder le fichier .iso en question dans le répertoire /root

5.1. Monter et configurer LTSP

```
#mkdir /mnt/ltsp

#mount -o loop /root/ltsp-4.iso /mnt/ltsp

#cd /mnt/ltsp
```

¹ <http://www.ltsp.org>

```
#!/tsp_installer
```

C'est ici que l'installation de LTSP s'effectue...

```
Make selection : 2
```

```
Make selection : 1
```

```
Make selection : A
```

```
Are you sure you want to install ALL components ? Yes
```

```
In which directory would you like to place LTSP ? [/opt/ltsp]
```

```
Installation de ltsp .....
```

Et c'est ici que la configuration de LTSP s'effectue...

```
#!/tspcfg
```

```
press enter to continue ...
```

```
Make selection : C
```

```
Make selection : 1
```

```
Select run level : 5
```

```
Make selection : 2
```

```
choisir eth2
```

```
Make selection : 8
```

```
Do you want to add entries to /etc/hosts ? Yes
```

```
Make selection : R
```

À présent, copions le noyau de ltsp dans le dossier diskless

```
#cp /tftboot/bzImageltsp /diskless/bzImageltsp
```

5.2. Modification finale au fichier exports

Éditons maintenant le fichier "exports" :

```
#nano /etc/exports
```

en ajoutant les lignes suivantes :

```
opt/ltsp/i386          192.168.0.0/255.255.255.0(ro,no_root_squash,async)
/var/opt/ltsp/swapfiles 192.168.0.0/255.255.255.0(rw,no_root_squash,async)
```

Voici la version finale du fichier exports, ou du moins, ce à quoi il devrait ressembler :

```
# /etc/exports: NFS file systems being exported.  See exports(5).

#diskless
/diskless/192.168.50.10 192.168.50.10(rw,no_root_squash,no_all_squash,sync)
/diskless/192.168.50.11 192.168.50.11(rw,no_root_squash,no_all_squash,sync)
/opt                    192.168.50.0/24(ro,no_root_squash,no_all_squash,sync)
/usr                    192.168.50.0/24(ro,no_root_squash,no_all_squash,sync)
/home                   192.168.50.0/24(rw,no_root_squash,no_all_squash,sync)
/var/log                192.168.50.10(rw,no_root_squash,no_all_squash,sync)
/var/log                192.168.50.11(rw,no_root_squash,no_all_squash,sync)

#ltsp
/opt/ltsp/i386          192.168.51.0/255.255.255.0(ro,no_root_squash,async)
/var/opt/ltsp/swapfiles 192.168.51.0/255.255.255.0(rw,no_root_squash,async)
```

5.3. Configuration du DHCP

DHCP signifie Dynamic Host Configuration Protocol. Le DHCP (service système) est un outil permettant à un ordinateur de se connecter sur un réseau et recevoir automatiquement une adresse IP de la part du serveur. Dans notre cas, les adresses sont ordinairement des adresses locales (192.168.x.x ou encore 10.0.x.x) et serviront strictement aux ordinateurs situés sur le réseau pour communiquer entre eux. L'adresse IP servant à la navigation sur Internet ne devrait être connue que par le serveur (interface eth0).

Éditons maintenant le fichier "dhcpd.conf" :

```
#nano /etc/dhcp/dhcpd.conf
```

Voici la version finale du fichier exports, ou du moins, ce à quoi il devrait ressembler :

```
ddns-update-style none;

option space PXE;
option PXE.mtftp-ip          code 1 = ip-address;
```

```
option PXE.mtftp-cport      code 2 = unsigned integer 16;
option PXE.mtftp-sport      code 3 = unsigned integer 16;
option PXE.mtftp-tmout      code 4 = unsigned integer 8;
option PXE.mtftp-delay      code 5 = unsigned integer 8;
option PXE.discovery-control code 6 = unsigned integer 8;
option PXE.discovery-mcast-addr code 7 = ip-address;

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

subnet 192.168.50.0 netmask 255.255.255.0
{
#diskless
    class "pxeclients"
    {
        match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
        option vendor-class-identifier "PXEClient";
        vendor-option-space PXE;

        option PXE.mtftp-ip 0.0.0.0;
        filename "pxelinux.0";
        next-server 192.168.50.1;
    }

    pool
    {
        max-lease-time 86400;
        default-lease-time 86400;
        deny unknown clients;
    }

    host dix
    {
        hardware ethernet      00:04:75:F4:E6:BC; //ici vous devez écrire
l'adresse MAC de votre propre carte réseau!
        fixed-address          192.168.50.10;
        server-name            "rouge";
        option routers         192.168.50.1;
        option domain-name-servers 192.168.0.14;
        option host-name        "dix";
    }
}
```

```
}

host onze
{
    hardware ethernet          00:04:75:F4:EC:07; //ici vous devez écrire
    l'adresse MAC de votre propre carte réseau !
    fixed-address              192.168.50.11;
    server-name                "rouge";
    option routers             192.168.50.1;
    option domain-name-servers 192.168.0.14;
    option host-name           "onze";
}
}

#ltsp

subnet 192.168.51.0 netmask 255.255.255.0
{
    option broadcast-address    192.168.51.255;
}

group {
    option domain-name         "nomdedomaine";
    filename                   "bzImageltsp";
    option root-path           "192.168.51.1:/opt/ltsp/i386";
    use-host-decl-names        on;

    host ws110 //Nous n'avons listé qu'un seul terminal pour l'exemple mais
               vous devriez lister tous les vôtres présents sur le réseau...
    {
        hardware ethernet      00:0A:E6:0D:00:93; //ici vous devez écrire
                                   l'adresse MAC de votre
                                   propre carte réseau!
        fixed-address           192.168.51.110;
    }
}
}
```

Après avoir quitté nano est enregistrant les modifications, démarrez le serveur DHCP :

```
#/etc/init.d/dhcp restart
```

5.4. Édition en vrac de quelques fichiers

- Fichier Xaccess : décommentez une ligne en enlevant le # au tout début :

```
#nano /etc/X11/xdm/Xaccess
```

```
# *                #any host can get a login window
```

- Fichier xdm-config : commentez la ligne DisplayManager.requestPort:0 (en ajoutant un # au tout début)

```
#nano /etc/X11/xdm/xdm-config
```

- Fichier kdmrc : cherchez la section [Xdmcp] et remplacez Enable = false par Enable = True

```
#nano /usr/kde/3.1/share/config/kdm/kdmrc
```

- Fichier gdm.conf : cherchez la section [xdmcp] et remplacez Enable = false par Enable = True

```
#nano /etc/X11/gdm/gdm.conf
```

- Fichier lts.conf : Pour y faire ??????????

```
#nano /opt/ltsp/i386/etc/lts.conf
```

- Sortez de nano en enregistrant les changement et entrez le lignes suivantes en console :

```
#rc-update add xdm default
```

```
#/etc/init.d/xdm start
```

5.5. Voilà!

Vous avez maintenant terminé l'installation de votre système! N'oubliez pas d'activer le démarrage sur la carte réseau dans le BIOS de vos terminaux X!

Vous devriez maintenant avoir un serveur, des postes diskless et des terminaux X prêts à fonctionner (quelle est la différence? :-)