26 août 2008

Démarrez sans disque avec PXE, Grub et NFS

Catégorie : News Tags : lmhs

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Les machines de récupération sont souvent incomplètes ou en partie défectueuses. L'un des éléments les plus susceptibles de s'user est le disque dur. Mais, celui-ci n'est pas forcément nécessaire pour faire fonctionner un système GNU/Linux.

Les fidèles lecteurs de longue date se souviendront sans doute d'un article paru dans l'un des premiers numéros de GNU/Linux Magazine France. Celui-ci détaillait la marche à suivre pour démarrer un système sans unité de stockage au travers d'un réseau.
Depuis la parution de cet article au début de l'année 1999 (le temps passe vite), les choses ont changé et ce hors-série est l'occasion rêvée de se pencher à nouveau sur le sujet.
La configuration matérielle requise pour cet article peut être considérée comme le « haut du panier » de la catégorie « machines recyclées ». En effet, nous traiterons ici de PXE, une des spécifications définies par le standard PC99.
Les machines trop anciennes ne supportent pas PXE et ne sont donc pas concernées par ce qui va suivre. Je traiterais brièvement, en annexe, d'une solution adaptée aux configurations plus anciennes en utilisant Etherboot.
Pour qu'une configuration supporte PXE, elle doit intégrer une interface réseau répondant à ces spécifications. On trouvera dans le commerce des cartes mère intégrant l'adaptateur réseau pour quelques 30 euros ou encore des adaptateurs réseau PCI dont le prix peut varier entre 50 et 100 euros. Enfin, dernière solution, si vous disposez d'un adaptateur réseau pouvant accueillir une bootrom, vous pouvez achetez une EEPROM contenant le code permettrant de démarrer en PXE. A titre indicatif, une carte D-Link DFE-530TX vous coutera quelques 10 euros.
Comptez autant pour une bootrom PXE compatible. Une carte 3Com 3C905C-TX-M intégrant PXE vous coûtera quelques 50 euros, neuve. Bien sûr, on trouvera pour un coût bien inférieur des cartes d'occasion sur les sites d'enchères en ligne.
Les spécifications PXE en elles-mêmes ne représentent qu'un élément de départ permettant de démarrer un système dit « diskless ». PXE est, grossièrement, une extension du protocole DHCP.
De ce fait, un serveur DHCP moderne et compatible sera suffisant pour répondre aux requêtes PXE en provenance des terminaux. Le support PXE sur le serveur permettra l'envoi d'un binaire au terminal via TFTP (Trivial FTP) qui l'exécutera. C'est ensuite ce binaire qui poursuivra la procédure de démarrage. Dans le cadre de cet article, c'est un binaire Grub que nous utiliserons.

DHCP

Nous utiliserons ici dhcp3-server qui offre toutes les fonctionnalités requises. Le système serveur est un GNU/Linux Debian Sarge. Les chemins et noms de fichiers décrits ici peuvent être sensiblement différents si vous utilisez une autre distribution.
~~dhcp3-server~~ se configure entièrement via le fichier ~~/etc/dhcp3/dhcpd.conf~~. Commençons pas les paramètres de configuration globaux ou communs à tous les hôtes :

```
# Aucun support DDNS
ddns-update-style none;

# De quel domaine nous occupons-nous ?
option domain-name „ed-diamond.com“;

# Quel est le serveur DNS ?
option domain-name-servers 80.10.246.130;

# La passerelle par défaut
option routers 192.168.0.10;

# Quelle facilité syslog utilisons-nous ?
log-facility local7;

# Quelques valeurs temporelles pour les
# bails des clients
default-lease-time 600;
max-lease-time 7200;

# Définition de notre sous réseau
subnet 192.168.0.0 netmask 255.255.255.0 {
}
```

Les commentaires parlent d'eux-mêmes. DHCP comme son nom l'indique (Dynamic Host Configuration Potocol) ne se limite pas aux fonctionnalités d'attributions d'a-dresses IP aux hôtes client.
Il est possible, via ce protocole, de communiquer un très grand nombre d'informations aux clients. La plupart des options de configuration sont définies par défaut.
C'est le cas, par exemple, de ~~domain-name~~, ~~domain-name-servers~~ ou encore de l'option ~~routers~~. Il s'agit d'options DHCP standards.
Ici, nous voulons que le serveur DHCP informe le client de sa configuration mais également qu'il lui transmette le binaire qu'il devra exécuter pour démarrer.
Notre choix s'étend porté sur Grub. Nous allons avoir besoin de créer une option supplémentaire afin de passer en argument l'emplacement du fichier menu.lst contenant la configuration de Grub.
Ceci n'est pas un fonctionnement « normal ». Habituellement, le binaire transmis aux client est statique et ne requiert aucune configuration particulière.
Pour créer une option, nous devons utiliser une ligne spécifique dans notre ~~dhcpd.conf~~ :

```
option grubmenu code 150 = string;
```

Chaque option DHCP possède un nom, un code et un type. Le nom est purement « pratique » et permet la manipulation simple des options dans la configuration.
Le code est utilisé dans la communication entre le serveur et le client et permet de se référer à l'option concernée. Le type renseigne sur la nature de l'information véhiculée par l'option. Ici nous utilisons le code ou tag 150.
Ceci n'est pas un choix arbitraire. C'est le code que Grub utilise pour récupérer l'information dont il a besoin. La nature de cette information est une chaîne de caractères désignant, au format Grub, l'emplacement du fichier de configuration ~~menu.lst~~.

Ceci fait, nous pouvons maintenant nous pencher sur le profil de notre futur client DHCP :

```
host lithium {
  hardware ethernet 00:30:1B:34:32:0F;
  fixed-address 192.168.0.38;
  option host-name „lithium“;
  server-name „raven“;
  filename „/mnt/six2/diskless/boot/pxegrub“;
  option grubmenu „(nd)/mnt/six2/diskless/boot/menu.lst“;
  option root-path „/mnt/six2/diskless/“;
}
```

Nous désignons le profil d'après le futur nom d'hôte du client, lithium.
Nous lui attribuerons une adresse IP fixe en fonction de l'adresse MAC de l'adaptateur Ethernet utilisé. Les éléments de configuration les plus importants ici sont :
~~filename~~ : référence le binaire que nous transmettrons au client afin qu'il puisse démarrer. ~~pxegrub~~ est un composant de Grub qu'il nous faudra obtenir par la suite.
~~grubmenu~~ est notre fameuse option ajoutée. nd dans la syntaxe propre à Grub désigne un network drive.
~~root-path~~ désigne le disque racine du client. Vous l'aurez compris, le système de fichiers racine de notre client sera placé

dans /mnt/six2/diskless/. C'est ici que nous devrons installer un système minimal par la suite.
Notre configuration DHCP est maintenant complète.
Si nous souhaitons offrir le même service à d'autres clients diskless, nous n'aurons qu'à créer d'autres profils similaires sur cette base.

PXE Grub

Le Grub livré en standard avec la distribution Debian Sarge ne met pas à disposition le chargeur pxegrub.
Nous allons donc devoir contourner le problème. Ici, plus d'une solution s'offre à nous :
Télécharger et compiler un Grub manuellement.
Récupérer un Grub complet en provenance d'une autre distribution et en extraire pxegrub.
Respecter la « manière Debian » en récupérant les sources Debian de Grub et en construisant un nouveau paquet.
C'est vers cette dernière solution que nous nous tournerons. Comprenez bien que c'est davantage par principe que pour des raisons techniques.
Nous n'avons, en principe, besoin que d'un seul binaire qui n'est aucunement lié au reste du système serveur.
Cependant lorsqu'il est question de faire intervenir des binaires dans un système, mieux faut tout mettre en œuvre pour garder une certaine homogénéité dans la distribution.
Nous allons donc récupérer les sources avec :

```
$ cd /tmp
$ apt-get build-dep grub
$ apt-get source grub
$ cd grub-0.95+cvs20040624
```

Par défaut, l'option permettant de construire les chargeurs réseau n'est pas utilisée au lancement du script configure.
Nous allons remédier à cela en éditant le fichier debian/rules. Repérez simplement l'étiquette configure-stamp dans le fichier.
Vous trouverez en dessous les lignes qui nous intéressent. Insérez ici les options nécessaires :

```
--enable-diskless --enable-rtl8139
```

Nous activons ici le pilote pour une carte Realtek 8139, car il s'agit du matériel présent dans le client. Vous pouvez obtenir la liste des pilotes disponibles à l'aide d'un simple ./configure --help | grep driver.
Vous pouvez également consulter le fichier netboot/README.netboot qui fournit de plus amples informations.
Ces modifications faites, assurez-vous d'être à la racine des sources du paquet et lancez la compilation/construction :

```
$ dpkg-buildpackage -rfakeroot -b
```

Un message vous signalera que le paquet ne peut être signé car vous ne disposez pas de la clef secrète du responsable du paquet (Jason Thomas).
Ceci n'empêchera en rien l'installation des nouveaux paquets fraîchement créés. Dans le répertoire supérieur, vous trouverez trois archives Debian (grub, grub-disk et grud-docs).
Assurez-vous de la présence du binaire qui nous intéresse et installez éventuellement le ou les paquets :

```
$ cd ..
$ dpkg --contents \
  grub_0.95+cvs20040624-17_i386.deb | \
  grep pxe
-rw-r--r-- root/root    122600 2005-05-28
    17:14:22 ./lib/grub/i386-pc/pxegrub

$ sudo dpkg -i grub_0.95+cvs20040624-17_i386.deb
Password:
[...]
Dépaquetage de la mise à jour de grub...
Paramétrage de grub (0.95+cvs20040624-17)...
```

Si vous avez choisi d'installer le paquet, vous retrouverez pxegrub dans /lib/grub/i386-pc.
Profitez-en pour le copier à l'emplacement que nous avons spécifié dans la configuration du serveur DHCP :

```
$ mkdir -p /mnt/six2/diskless/boot
$ cp /lib/grub/i386-pc/pxegrub \
  /mnt/six2/diskless/boot
```

Il ne nous reste plus, en ce qui concerne Grub, qu'à créer un fichier ~~menu.lst~~ au bon emplacement :

```
timeout 15
default 0
title  diskless
root (nd)
kernel /mnt/six2/diskless/boot/vmlinuz rw ip=dhcp root=/dev/nfs nfsroot=192.168.0.1:/mnt/six2/diskless
initrd /mnt/six2/diskless/boot/initrd.img
```

On retrouve le (nd) permettant à Grub de récupérer en TFTP l'image du noyau et celle du RAMdisk de démarrage.
On précise en argument du noyau l'obtention d'une adresse IP via DHCP et un système de fichier racine en NFS provenant d'un serveur utilisant l'adresse 192.168.0.1.
Celui-ci, pour l'heure encore non configuré, est également notre serveur DHCP, mais ce n'est pas une obligation.

Le serveur TFTP

Nous n'avons pas encore mis en place de système permettant aux clients de récupérer les images binaires.
Pour ce faire, nous devons déployer un serveur TFTP. Il en existe plusieurs répondant à différents critères. Nous avons opté pour ~~atftpd~~ pour sa facilité de configuration entièrement guidée par le système de configuration Debian.
Il n'existe pas, à ma connaissance de raisons suffisantes pour faire fonctionner un tel serveur en mode démon.
L'utilisation d'~~inetd~~ ou ~~xinetd~~ est donc suffisante. Si, comme moi, vous avez une nette préférence pour xinetd, ajoutez simplement un fichier dans votre ~~/etc/xinted.d~~ :

```
service tftp
{
 socket_type     = dgram
 protocol        = udp
 wait            = no
 user            = nobody
 server          = /usr/sbin/in.tftpd
 server_args     = --tftpd-timeout 300 --no-multicast --maxthread 5 --verbose=5 /mnt/six2/diskless/boot
}
```

TFTP est un protocole utilisant UDP. Les quelques paramètres passés en argument du binaire ~~in.tftpd~~ ne sont pas critiques en dehors, bien sûr, du dernier qui est l'emplacement du dépôt.
Vous pourrez facilement tester le fonctionnement du serveur en installant un client TFTP (paquet ~~tftp~~) et en récupérant un fichier :

```
$ cd /tmp
$ tftp 127.0.0.1
tftp>get pxegrub
Received 123402 bytes in 0.0 seconds
tftp> quit
```

Vous pouvez également choisir le paquet ~~tftpd-hpa~~. Voici l'entrée ~~xinetd~~ correspondante :

```
service tftp
{
 socket_type     = dgram
 protocol        = udp
 wait            = yes
 user            = root
 server          = /usr/sbin/in.tftpd
 server_args     = -s /mnt/six2/diskless/boot
}
```

Attention ! En fonction du serveur TFTP et de sa configuration, les chemins ne peuvent être spécifiés de manière absolue mais relativement à la racine du dépôt TFTP, pour les fichiers ~~pxegrub~~, ~~menu.lst~~ et les éléments du noyau. Si vous rencontrez des problèmes et des messages concernant des fichiers non trouvés, c'est de ce coté qu'il faudra chercher.
Si vous désirez lancer ce serveur en mode démon, il ne faudra pas oublier de changer le fichier ~~/etc/default/tftpd-hpa~~ de manière à correspondre avec l'emplacement de votre dépôt :

```
#Defaults for tftpd-hpa
RUN_DAEMON="no"
OPTIONS="-l -s /mnt/six2/diskless/boot"
```

Le serveur NFS

Après le démarrage du client, la réception et l'exécution de Grub, le téléchargement des images noyau et d'~~initrd~~, le système aura besoin d'un véritable système de fichiers.
Celui-ci sera monté à la racine et utilisé comme avec un disque local. Pour faire cela, nous allons utiliser NFS. Notez qu'il est possible d'utiliser d'autres systèmes de fichiers en réseau, mais nous resterons dans la configuration la plus classique ici.
Nous avons besoin d'un serveur NFS. Ceci s'installera très facilement et très rapidement via le paquet ~~nfs-kernel-server~~.
Bien sûr, vous pouvez également choisir ~~nfs-user-server~~ offrant plus de fonctionnalités mais étant plus « coûteux » en termes de ressources.
Côté configuration, le serveur NFS n'est pas très exigeant, une simple ligne dans ~~/etc/exports~~ et le tour est joué :

```
/mnt/six2/diskless 192.168.0.0/255.255.255.0(ro,no_root_squash,async)
```

On limite ici les accès à un sous-réseau spécifique, mais il est parfaitement possible d'être plus (ou moins) restrictif.

Un système pour le terminal : simple et rapide

Tout est maintenant en place pour créer un système racine pour notre client. Il nous faut un noyau, une image initrd et un système de fichiers peuplé et viable. La solution de facilité ici est de faire appel au paquet ~~lessdisks~~.
Lessdisks est un projet basé sur la distribution Debian et visant à créer simplement un système de fichiers pour des clients diskless.
L'installation est entièrement guidée via ~~debconf~~. Il suffit donc d'installer le paquet avec la commande ~~apt-get~~. Un certain nombre de questions vous seront posées à ce moment-là.
Une fois le paquet installé, vous n'aurez plus qu'à utiliser la commande ~~lessdisks-install~~ en tant qu'utilisateur root.
Passez par le menu Advanced Configuration pour pouvoir choisir toutes les options et en particulier l'emplacement du système de fichiers racine pour vos clients diskless.
lessdisks-install après configuration vous permettra d'installer un système en utilisant des paquets Debian.
Préférez donc des dépôts qui sont les mêmes que pour votre système installé et n'oubliez pas de configurer votre serveur proxy afin d'optimiser les téléchargements.
Vous remarquerez à un moment de la procédure de configuration qu'il vous est demandé un nom de paquet noyau pour les machines client. J'y reviendrai par la suite mais sachez que vous ne pouvez pas utiliser un noyau standard.
Il faut, en effet, que ce dernier puisse démarrer sur un système de fichiers NFS. Le paquet ~~kernel-image-netbootable~~ proposé par défaut est donc le choix optimal dans de nombreux cas.
Notre client diskless ne pourra pas utiliser le montage NFS en lecture/écriture. Cependant, un système GNU/Linux à besoin de posséder des droits en écriture pour un certain nombre de répertoires (~~/tmp~~, ~~/var/*~~, etc.). Pour régler le problème, on utilisera des systèmes de fichiers en mémoire.
Vous aurez le choix en ce qui concerne le type de système de fichiers durant la procédure de configuration.
Une fois l'ensemble correctement configuré assurez-vous que votre actuel système soit à jour et en particulier le paquet ~~debootstrap~~.
C'est lui qui installera les paquets essentiels au nouveau système, s'il n'est pas à jour, il tentera d'installer des paquets qui ne sont plus disponibles dans les dépôts.
Ceci fait, passez simplement pas l'entrée Begin Installation et trouvez quelque chose d'intéressant à faire en attendant ;)
Au terme du téléchargement des paquets et de leur installation, différentes questions supplémentaires vous seront posées.
Ceci concerne principalement l'interface graphique et la configu-ration automatique du matériel. Le projet Lessdisks a pour but de créer un système le plus générique possible et le plus auto-configurable.
Il vous sera toujours possible, par la suite de personnaliser l'installation.
Cependant, si vous souhaitez obtenir une configuration sur mesure, mieux vaut alors se pencher sur une autre technique de construction du système racine comme LFS, par exemple.
Une fois l'installation terminée, votre répertoire se sera vu ajouté quelques 300 Mo.
Vous retrouverez alors les éléments qui nous manquaient :

```
$ ls -F /mnt/six2/diskless/boot
 config-2.4.27-2-386
 initrd.img@
 initrd.img-2.4.27-2-386
 menu.lst
 pxegrub
 System.map-2.4.27-2-386
 vmlinuz@
 vmlinuz-2.4.27-2-386
```

Vérifiez une dernière fois que tout est en ordre dans votre configuration serveur et en particulier les éléments concernant le démarrage avec Grub et démarrez le poste client sur PXE.
Après mise sous tension, le BIOS de votre carte réseau va faire une requête DHCP et notre serveur lui répondra et lui transmettra ~~pxegrub~~.

Celui-ci, à son tour récupérera l'~~initrd~~ et l'image du noyau et leur passera la main. Grâce aux arguments passés, le noyau fera une autre requête DHCP, se configurera et tentera de monter la racine du système via NFS.
Vous allez sans doute remarquer un certain nombre de messages d'erreur, principalement en rapport avec le fait que le montage se faire en lecture seule.
Ceci est certes désagréable mais pas réellement problématique. En revanche, s'il est question de problème de connexion réseau, c'est un peu plus grave.
pxegrub dispose d'un certain nombre de pilotes pour les interfaces réseau, mais le noyau a les siens.
Il est possible, mais rare, que certains chipsets (realtek par exemple) soient détectés et utilisés par Grub mais pris en charge par le noyau directement. C'est le cas, par exemple d'une des cartes testées à base de rtl8169.
Fort heureusement, vous pouvez, presque à loisir, modifier les éléments de démarrage de Lessdisks. C'est le cas, par exemple pour l'image ~~initrd~~ utilisée au démarrage. Pour modifier les modules et les paramètres utilisés pour trouver l'interface réseau, il vous suffit d'éditer le fichier ~~/etc/mkinitrd/mkinitrd.conf~~ du nouveau système.
Là, nous avons ajouté le module ~~r8169~~ dans la liste à la ligne :

```
nic_modules="ne 3c509 3c59x..."
```

Vous pouvez également gagner en vitesse de démarrage en retirant des modules de la liste. Un autre fichier qu'il vous est possible de modifier dans la même intention est ~~/etc/mkinitrd/mkinitrd.conf~~.
Il s'agit du fichier de configuration de ~~mkinitrd~~. Là, vous pouvez jouer sur l'option ~~MODULES~~ afin de réduire la taille de l'image finale et ainsi gagner en temps de transfert TFTP.
Cette option décide de l'embarquement de modules kernel dans l'image ~~initrd~~. Les valeurs possibles sont ~~all~~, ~~most~~, ~~dep~~ ou ~~none~~. Celles-ci parlent d'elles-mêmes.
Une fois un changement fait dans l'un des ces fichiers, n'oubliez pas de régénérer l'image avec :

```
$ lessdisks-chroot dpkg-reconfigure \
  kernel-image-2.4.27-2-386
```

~~lessdisks-chroot~~ est un script très intéressant qui sert de wrapper afin d'agir, avec les outils Debian, sur le système pour le client.
Vous pouvez ainsi reconfigurer n'importe quel élément du système. On en profitera pour passer notre clavier en français au démarrage.

```
$ lessdisks-chroot dpkg-reconfigure \
  console-data
```

Lessdisks est un vaste projet. Installez le paquet de documentation et étudiez le fonctionnement des différents scripts. Ceci vous permettra de tirer le meilleur de l'ensemble et, pourquoi pas, de participer au développement.

Etherboot

PXE est quelque chose de relativement récent. Avant l'utilisation de ce standard, les cartes réseau supportant une EEPROM pouvaient se voir ajouter un code pour démarrer sur le réseau.
Il existait ainsi des EEPROM ou bootroms pour démarrer au travers le réseau de la même manière qu'avec PXE. Etherboot est un projet de longue date (mais toujours pleinement en vie) permettant de créer ce type d'EEPROM.
On peut ainsi, sous réserve de disposer du matériel adéquat, compiler et programmer ses propres bootroms pour ses interfaces réseau.
La configuration d'un serveur pour Etherboot est très proche de ce que nous venons de voir (DCHP/TFTP/NFS). On notera également qu'il sera possible, sans carte avec EEPROM de procéder de même en utilisant un support disquette.
Enfin, signalons que le projet Etherboot implémente actuellement PXE. La version CVS permet d'ores et déjà de créer ses propres bootroms PXE pour quelques interfaces du marché.
Arrêtons-nous là pour cette fois. J'espère que cet article vous aura donné l'eau à la bouche et vous aura décidé à réveiller ces vielles machines qui dorment dans un coin.
Et pourquoi ne pas en faire un cluster ? Mais c'est une autre histoire...

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Posté par Denis Bodor (Lefinnois) | Signature : Denis Bodor | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- La protection du secret : approche juridique
- Mieux connaître OOo Draw : les objets 3D et leur espace
- Calc et les variables
- Noël 94 : le cas Mitnick-Shimomura ou comment le cyber-criminel a souhaité joyeux Noël au samurai
- Mettre en place une politique SSI : des recettes pratiques
- Extensions de Firefox : notre sélection
- Konversation : pour discuter librement sur IRC
- BitTorrent : l'autre façon d'échanger des fichiers
- Au-delà de Diffie-Hellman … ?
- Envy : l'installation facile des drivers graphiques dernier cri pour ATI et Nvidia

Il y a actuellement

- 743 articles/billets en ligne.

Recherche

Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution

- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

Archives

- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 108
- GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 38
- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°49
- À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49
- Linux Pratique Essentiel N°3 - Août/Septembre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°3

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse
- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux