LTSP − Linux Terminal Server Project − v4.1

James McQuillan

<jam@LTSP.org>

Pierre Baco

Traduction française

<pbaco@carlit.net>

Historique des versions

Version 4.1	2004−06−20	Revu par : jam
Version 4.1.3−0−fr	2004−06−20	Revu par : jam

GNU/Linux est une plate−forme idéale pour le déploiement de stations sans disque ou "clients légers". L'objectif premier de ce document est d'expliquer comment déployer des clients légers en utilisant LTSP. Ce document couvre également plusieurs autres éléments relatifs aux clients légers et stations de travail.

<!−− TOCTITLE="Table des Matières" −−>

<!−− TOCTITLE="Table des Matières" −−>

Introduction

Le Projet de Serveur de Terminaux Linux (LTSP) offre une solution simple pour utiliser des micro−ordinateurs peu coûteux (ou recyclés) comme des terminaux graphiques ou caractère d'un serveur GNU/Linux.

Dans un environnement classique, on trouve des micro−ordinateurs à base Intel sur chaque bureau, disposant de plusieurs giga−octets d'espace disque. Les utilisateurs y stockent leurs données respectives, mais rares sont les disques durs locaux effectivement sauvegardés.

Est−il vraiment raisonnable d'installer un ordinateur complet sur chaque bureau ?

Nous pensons que non.

Il existe heureusement une autre solution. En utilisant LTSP, vous pourrez (ré)utiliser des PCs d'entrée de gamme, en retirer le disque dur, le lecteur de disquette et de CD−ROM et y installer une carte réseau. La plupart des cartes réseau disposent d'un emplacement prévu pour accueillir un "bootrom" (un circuit contenant un programme de démarrage), qu'il suffit d'insérer pour démarrer depuis le réseau.

Pendant la phase de démarrage, la station sans disque (client léger) récupère son adresse IP et un noyau Linux depuis le serveur, puis monte la racine de son système de fichiers (/) depuis ce même serveur via NFS.

Le client léger peut être configuré dans un des 3 modes suivants :

- **Interface graphique X Window**

  Avec X Window, le client léger peut être utilisé pour accéder à n'importe quelle application installée sur le serveur LTSP, ou n'importe quel autre serveur du même réseau.

- **Sessions Telnet en mode caractère**

  Le client léger peut créer plusieurs sessions telnet vers le serveur. Chacune de ces sessions telnet sera accessible sur un écran virtuel séparé. En utilisant les touches Alt−F1 à Alt−F9, vous pouvez ainsi passer d'une session telnet à une autre.

- **Ligne de commande (prompt shell)**

  Le client léger peut enfin être configuré pour entrer directement dans un shell bash en mode console locale. Ceci est particulièrement utile pour la mise au point en cas de problème avec X Window ou NFS.

Le grand avantage de LTSP est de pouvoir déployer un nombre important de clients légers à partir d'un seul serveur GNU/Linux. Combien ? Cela dépend bien sûr des capacités du serveur et des applications qui seront utilisées.

Il est tout à fait possible de faire fonctionner Mozilla et Open Office sur 50 clients légers reliés au même serveur bi−processeur P4 2.4 Ghz avec 4 Go de Ram. Nous savons que cela marche. Et la charge moyenne du serveur dépasse rarement 1.0 !

---

1. Limite de responsabilité

En aucun cas l'auteur, le(s) distributeur(s) et toute autre personne ayant contribuée à ce document ne peuvent être tenus responsables pour les dommages physiques, financiers, moraux ou autres qui pourraient survenir par la mise en pratique des recommandations exposées dans ce texte.

2. Droit d'auteur (Copyright) et Licence

Chapitre 1. Principes de fonctionnement

L'initialisation et le démarrage (boot) d'un client léger s'effectue en plusieurs étapes. Si un problème devait apparaître, une bonne connaissance de chacune de ces étapes facilitera grandement la mise au point et la correction du problème. Ce chapitre décrit en détail ce processus de démarrage.

Pour démarrer un client léger, quatre services de base sont nécessaires sur le serveur. Ce sont :

- DHCP
- TFTP
- NFS
- XDMCP

La configuration de LTSP est très souple : ces services peuvent être fournis par le même serveur, ou par des serveurs différents. Pour l'exemple qui suit, nous utiliserons une configuration simple, dans laquelle tous les services sont fournis par le même serveur.

1.1. Etapes de démarrage du client léger

1. Chargement du noyau Linux en mémoire du client léger. Ceci peut être réalisé de plusieurs manières :

   a. avec un Bootrom (Etherboot,PXE,MBA,Netboot) ;
   b. à partir du lecteur de disquette ;
   c. à partir du disque dur ;
   d. depuis un CD−ROM ;
   e. ou d'une clé mémoire USB.

   Chacune de ces méthodes de démarrage sera détaillée dans ce chapitre.

2. Une fois le noyau Linux chargé dans la mémoire du client léger, il va commencer à s'exécuter.

3. Le noyau va initialiser l'ensemble du système et tous les périphériques qu'il peut reconnaître.

4. C'est ici que les choses sérieuses commencent : pendant le chargement du noyau, une image de disque virtuel est également chargée en mémoire du client léger. Un argument de la ligne de commande **root=/dev/ram0** demande au noyau de monter cette image comme racine de son système de fichiers.

5. En principe, lorsque le noyau a terminé son démarrage, il lance le programme **init** pour continuer l'initialisation du système. Mais ici, il est demandé au noyau de lancer un petit script shell se trouvant sur le disque virtuel. Ceci est fait en passant l'argument **init=/linuxrc** sur la ligne de commande du noyau.

6. Le script **/linuxrc** commence par un balayage (scan) du bus PCI du client léger, afin de rechercher une carte réseau. Chaque composant PCI trouvé est comparé aux descriptions stockées dans le fichier `/etc/niclist`. En cas d'égalité, le nom du module−driver correspondant à la carte réseau trouvée est renvoyé par le script, et ce module est chargé en mémoire. Pour les cartes ISA (qui ne sont pas détectées automatiquement), le nom du module−driver DOIT impérativement être spécifié sur la ligne de commande du noyau, ainsi que ses paramètres éventuels (IRQ, adresses E/S, DMA, etc).

7. Un petit module client DHCP appelé **dhclient** est ensuite lancé, afin d'envoyer une nouvelle requête

au serveur DHCP. Cette requête, émise depuis le mode user, est nécessaire pour récupérer des informations supplémentaires que la première requête du bootrom ne pouvait retourner.

8. Lorsque **dhclient** reçoit une réponse du serveur, il exécute alors le script **/etc/dhclient−script**, qui récupère les informations renvoyées et configure l'interface eth0.

9. Jusqu'à cette étape, le système de fichiers était un disque virtuel en ram. Le script **/linuxrc** va maintenant monter un nouveau système de fichiers sur sa racine (root) via NFS. Le répertoire exporté par NFS depuis le serveur vers le client léger est en principe `/opt/ltsp/i386`. Cette opération ne peut être réalisée en montant directement ce répertoire sur `/`. Il doit d'abord être monté sur `/mnt`, puis la commande **pivot_root** est lancée. **pivot_root** va échanger la racine courante (le disque virtuel) avec le nouveau système de fichiers. Lorsque cela est terminé, le répertoire exporté par NFS est monté sur `/` et l'ancienne racine est montée sur `/oldroot`.

10. Une fois le montage et l'échange de racine effectués, le script **/linuxrc** a terminé son travail. Il est temps d'appeler le véritable programme **/sbin/init**.

11. Init va lire le fichier `/etc/inittab` et mettre en place l'environnement du client léger.

12. Un des premières commandes d'`inittab` à être exécutée est le script **rc.sysinit** qui restera actif tant que le client léger est dans sa phase '***sysinit***'.

13. Le script **rc.sysinit** crée un nouveau disque virtuel en mémoire (ramdisk) de 1 Mo, destiné à contenir toutes les données et informations devant être écrites ou modifiées par la suite (disque en lecture/écriture).

14. Ce disque virtuel est monté en tant que `/tmp` sur le système de fichiers local du client léger. Tous les fichiers en lecture/écriture du système de fichiers seront stockés dans `/tmp`, et ce même s'ils sont dans d'autres répertoires locaux, grâce à des liens symboliques pointant vers ces fichiers.

15. Le (pseudo) système de fichiers `/proc` est ensuite monté.

16. Le fichier `lts.conf` est examiné, et tous les paramètres spécifiques au client léger en cours d'initialisation sont exportés comme variables d'environnement, afin que le script **rc.sysinit** puisse les utiliser par la suite.

17. Si le client léger est configuré pour accéder à un fichier−partition de swap via NFS, le répertoire `/var/opt/ltsp/swapfiles` du serveur est monté sur le répertoire `/tmp/swapfiles` du client léger. Si le fichier de swap n'existe pas encore dans ce répertoire, il est automatiquement créé. Sa taille est paramétrable (à spécifier dans le fichier `lts.conf` du serveur).

    Le fichier de swap est alors activé par la commande **swapon**.

18. L'interface réseau ***loopback*** est ensuite configurée. Il s'agit de l'interface "interne" ayant pour adresse IP *127.0.0.1*.

19. Si l'exécution locale d'applications est activée, le répertoire `/home` du serveur est exporté via NFS pour être monté sur le répertoire `/home` du client léger, afin que les applications puissent avoir accès aux répertoires personnels des utilisateurs.

20. Plusieurs sous−répertoires sont créés dans `/tmp` pour stocker les fichiers temporaires nécessaires au fonctionnement du système. Les sous−répertoires comme :

    a. `/tmp/compiled`

    b. `/tmp/var`

    c. `/tmp/var/run`

    d. `/tmp/var/log`

    e. `/tmp/var/lock`

    f. `/tmp/var/lock/subsys`

    seront donc créés.

21. Le fichier `/tmp/syslog.conf` est créé. Ce fichier contient les paramètres nécessaires au fonctionnement du daemon **syslogd**, afin d'indiquer vers quel serveur du réseau le client léger doit envoyer ses messages à journaliser. Le nom du serveur syslog est spécifié dans le fichier `lts.conf`. Il existe un lien symbolique `/etc/syslog.conf` (position habituelle de `syslog.conf`) qui pointe vers `/tmp/syslog.conf`.

22. Le daemon **syslogd** est lancé, avec le fichier de paramètres créé à l'étape précédente.

23. Lorsque le script **rc.sysinit** est terminé, le contrôle revient au programme **/sbin/init**, qui va alors changer le niveau d'exécution (runlevel) de ***sysinit*** à ***5***.

    Ceci aura pour effet de lancer toutes les autres commandes spécifiées dans le fichier `/etc/inittab`.

24. Par défaut, le fichier `inittab` contient des lignes demandant l'exécution du script **/etc/screen_session** sur tty1, tty2 et tty3. Il est donc possible d'utiliser 3 sessions simultanées sur le client léger. Le type de ces sessions est contrôlé par les paramètres ***SCREEN_01***, ***SCREEN_02*** et ***SCREEN_03*** que l'on trouve dans le fichier de configuration `lts.conf`.

    Des lignes supplémentaires peuvent être ajoutées dans `inittab`, si d'autres sessions sont nécessaires.

25. Si le paramètre ***SCREEN_01*** de `lts.conf` a pour valeur ***startx***, le script **/etc/screen.d/startx** est exécuté, ce qui lancera l'interface graphique X Window sur le client léger.

    Dans le fichier **lts.conf**, il existe un paramètre appelé **XSERVER**. Si ce paramètre est absent, ou a pour valeur "**auto**", le script **startx** tente de détecter automatiquement quelle est la carte vidéo du client léger. Si la carte est de type PCI ou AGP, l'identifiant PCI de la carte (et de son fabricant) sera récupéré et comparé aux identifiants contenus dans le fichier `/etc/vidlist`.

    Si la carte vidéo est supportée par Xorg 6.7, la routine **pci_scan** renverra le nom du module−driver correspondant. Si la carte est seulement supportée par XFree86 3.3.6, **pci_scan** renverra le nom du serveur X à utiliser. Le script **startx** sait faire la différence entre les deux systèmes car le nom des anciens serveurs X 3.3.6 commence toujours par 'XF86_', tandis que le nom des modules du nouveau

serveur Xorg est en minuscules, comme par exemple *ati* ou *trident*.

26. Si Xorg est utilisé, le script **/etc/build_x4_cfg** est appelé pour créer un fichier `XF86Config` correspondant à la carte vidéo trouvée. Si XFree86 3.3.6 est utilisé, le script **/etc/build_x4_cfg** est appelé pour créer ce fichier. Ces fichiers sont alors stockés dans le répertoire `/tmp`, qui, on s'en souvient, est un disque virtuel (ramdisk), accessible seulement depuis et par le client léger.

    Le fichier `XF86Config` est également créé en fonction des paramètres trouvés dans le fichier de configuration `/etc/lts.conf`.

27. Une fois le fichier XF86Config créé, le script **startx** lance le serveur X correspondant avec ce nouveau fichier de configuration.

28. Le serveur X lance alors une requête ***XDMCP*** au serveur LTSP, qui ouvre un écran de login.

29. Arrivé à ce stade, l'utilisateur peut s'identifier et se connecter (login) au serveur. Une fois authentifié, l'utilisateur a enfin accès à une nouvelle session graphique sur le serveur, et à tous les programmes installés sur ce dernier.

    Au premier abord, ceci est un peu déroutant. L'utilisateur est assis face au client léger, mais tout le travail se déroule sur le serveur. Tous les programmes sont exécutés sur le serveur, et seule leur sortie est affichée sur l'écran du client léger.

1.2. Téléchargement du noyau en mémoire du client léger.

Le téléchargement du noyau Linux dans la mémoire du client léger peut être réalisé de plusieurs façons :

- boot ROM
- média local

1.2.1. Boot ROM

- Etherboot

  Etherboot est un projet de bootrom open−source (libre) très connu. Il contient les drivers de nombreuses cartes réseau, et fonctionne parfaitement avec LTSP.

  Pour pouvoir être téléchargés via Etherboot, les noyaux Linux doivent être "marqués" avec l'utilitaire **mknbi−linux**, qui prépare le noyau pour un démarrage (boot) par le réseau. Du code supplémentaire est inséré au début du noyau, tandis que le fichier `initrd` est ajouté à la fin.

  Les noyaux fournis avec LTSP sont déjà marqués et prêts à être téléchargés par Etherboot.

  Le code Etherboot peut aussi être stocké sur une disquette. Ceci est très pratique lorsqu'on n'a pas la possibilité d'intégrer Etherboot dans une PROM, ou simplement pour effectuer des tests avant de programmer une PROM.

- PXE

A l'origine, PXE est une partie de la spécification 'Wired for Management' datant des années 90, décrivant une technologie de bootrom appelée *Pre−boot Execution Environment*, plus communément abrégée en ***PXE***.

Un bootrom PXE peut télécharger un fichier de 32 Ko maximum. Mais un noyau Linux est nettement plus gros. En conséquence, LTSP est configuré pour télécharger un "boot−loader" (chargeur) de 2ème niveau appelé ***pxelinux***. pxelinux est assez petit pour être téléchargé par PXE, pour ensuite (2ème niveau) télécharger lui même des fichiers bien plus gros, comme un noyau Linux.

- MBA

Managed Boot Agent (MBA) est un bootrom conçu par la société ***emBoot***. A l'origine, emBoot était le département "Lanworks" de 3Com. MBA supporte en fait 4 bootroms en un seul. Il peut gérer PXE, TCP/IP, RPL et Netware.

L'implémentation PXE de MBA fonctionne parfaitement. Il est possible de l'utiliser avec pxelinux pour télécharger un noyau Linux.

L'option ***TCP/IP*** de MBA peut aussi être utilisée, mais le noyau doit subir une préparation spéciale, avec l'utilitaire **imggen**.

- Netboot

Netboot, comme Etherboot, est un autre projet libre d'images bootrom. Sa particularité est de construire un bootrom en "encapsulant" les drivers NDIS ou les packet drivers originaux fournis avec les cartes réseau.

1.2.2. Média local

- Disquette

Il y a 2 façons de démarrer un client léger LTSP à partir d'un lecteur de disquette. La première consiste à copier le code d'EtherBoot dans le secteur de démarrage d'une disquette. Une fois chargé, il fonctionnera exactement comme une véritable Boot PROM installée sur la carte réseau. Cette dernière sera initialisée, puis le noyau Linux sera chargé depuis le serveur LTSP.

Il est également possible de copier la totalité d'un noyau Linux et son fichier initrd sur une disquette, et de démarrer à partir de celle ci. Toutefois, et à condition que le noyau contienne sur la disquette, le chargement sera bien plus lent qu'avec la première méthode (par le réseau).

- Disque dur

Le même noyau (et son initrd) peut être installé sur un disque dur local, dont le secteur de démarrage (MBR) aura été initialisé avec LILO ou GRUB. Il est aussi possible de simplement copier le code Etherboot sur le secteur de démarrage du disque dur, qui agira alors exactement comme une PROM de boot ou une disquette.

- CD−ROM

Si le client léger permet de démarrer depuis le lecteur de CD−ROM, un CD−ROM bootable peut être utilisé, contenant soit un noyau Linux entier (et son initrd) ou le code Etherboot.

- Clé mémoire USB

  Tout comme un CD−ROM, une disquette ou un disque dur, une clé mémoire USB peut être utilisée pour démarrer un client léger (si le BIOS le supporte). Selon sa taille, la clé mémoire USB peut héberger le code Etherboot ou un noyau Linux complet et son fichier initrd.

Chapitre 2. Installation de LTSP sur le serveur

On peut considérer LTSP comme une distribution Linux complète, à ceci près qu'il s'agit d'une distribution installée "par dessus" la distribution déjà installée sur un serveur hôte. Ce serveur hôte peut utiliser n'importe quelle distribution Linux. De fait, il n'y a pas d'obligation stricte à ce que le serveur fonctionne sous Linux. Le seul pré−requis est que le serveur puisse être un serveur NFS (Network File System). La majorité des systèmes Unix propose cette fonctionnalité. Et de fait, certaines versions de Windows pourraient être configurées pour héberger un serveur LTSP.

La mise en place d'un serveur LTSP se déroule en 3 étapes.

- Installation des utilitaires LTSP.
- Installation des packages clients LTSP (pour les clients légers).
- Configuration des services requis par LTSP.

2.1. Installation des utilitaires LTSP

Depuis sa version 4.1, LTSP dispose d'un package d'utilitaires pour installer (télécharger) et gérer tous les autres packages LTSP (tout ce qui concerne les clients légers), et pour configurer les services sur le serveur LTSP.

L'utilitaire d'administration s'appelle **ltspadmin** et celui de configuration s'appelle **ltspcfg**. Ces deux programmes font partie du package **ltsp−utils**.

Le package ***ltsp−utils*** est disponible soit en format ***RPM***, soit en format ***TGZ*** (tar compressé). Vous pouvez choisir le format qui vous convient et suivre les instructions d'installation suivantes.

2.1.1. Installation du package RPM

Téléchargez la dernière version du package RPM **ltsp−utils** et installez la en utilisant la commande suivante:

```
rpm −ivh ltsp−utils−0.1−0.noarch.rpm
```

Cette commande installera les utilitaires sur le serveur.

2.1.2. Installation du package TGZ

Téléchargez la dernière version du package TGZ **ltsp−utils** et installez la en utilisant les commandes suivantes:

```
tar xzf ltsp−utils−0.1−0.noarch.tgz
cd ltsp_utils
./install.sh
cd ..
```

Ces commandes installeront les utilitaires sur le serveur. Ce format de package peut être utile pour les distributions Linux n'utilisant pas le format RPM (ex Debian).

2.2. Installation des packages clients LTSP

Une fois l'installation du package **ltsp−utils** terminée, on peut alors lancer la commande **ltspadmin**. Cet utilitaire se charge de récupérer tous les autres packages nécessaires, en interrogeant le site de téléchargement de LTSP pour en obtenir une liste à jour.

En lançant la commande **ltspadmin** on obtient l'écran suivant:

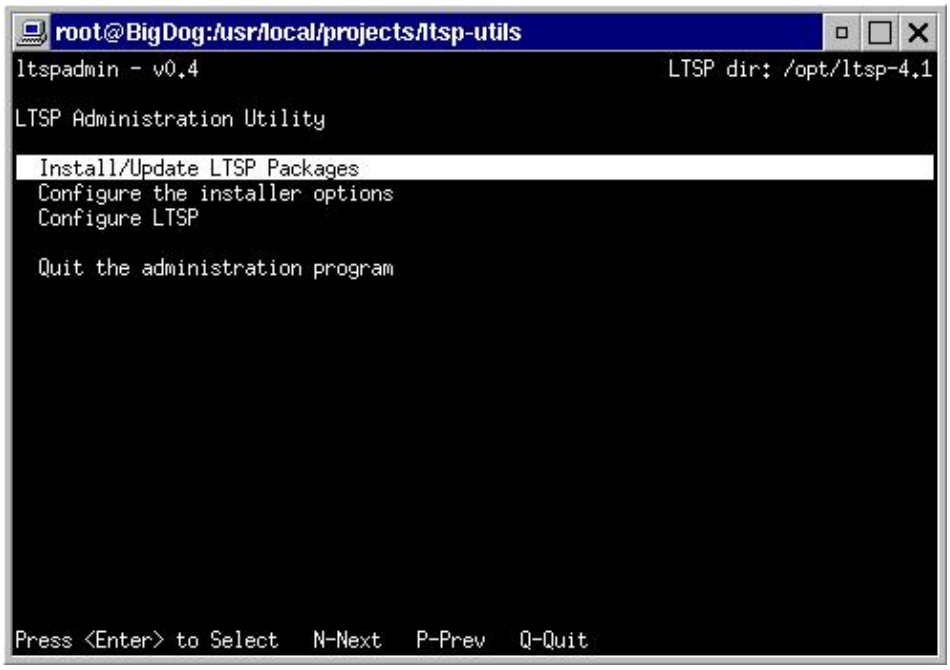

Figure 2−1. Installation de LTSP − Ecran principal

A partir de ce menu, choisissez l'option "Install/Update". Si le programme est lancé pour la première fois, l'écran de configuration suivant apparaît:

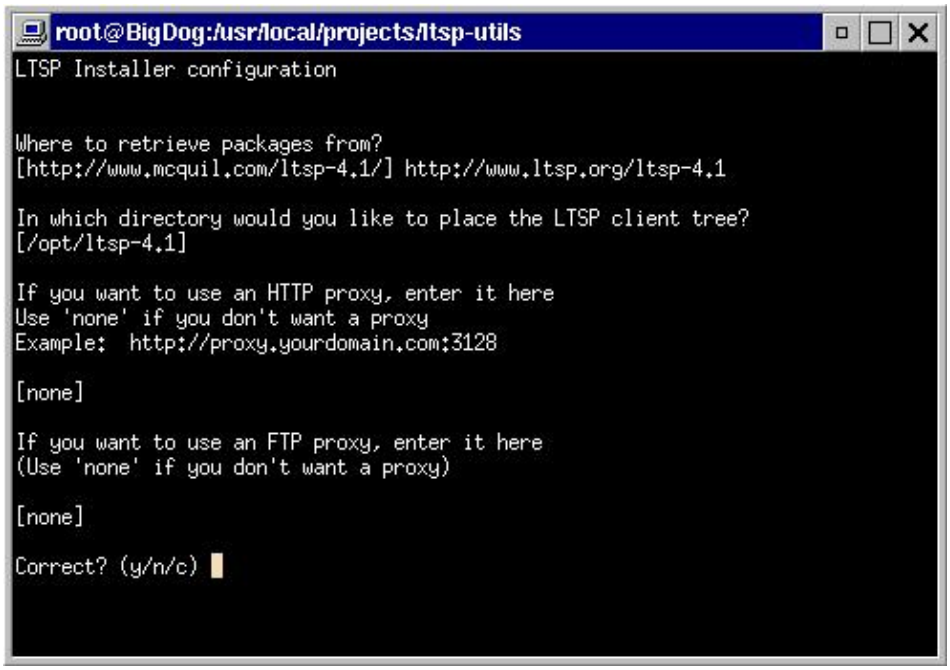

Figure 2−2. Installation de LTSP − Ecran de configuration

Dans l'écran ***configuration***, vous pouvez initialiser certains paramètres utilisés par le programme d'installation pour télécharger et installer les autres packages LTSP. Ces paramètres sont:

Where to retrieve packages from
: (Où récuperer les packages). Il s'agit d'une URL qui pointe vers le serveur de dépôt des packages. Par défaut, il s'agit de `http://www.ltsp.org/ltsp-4.1`, mais si vous souhaitez installer les packages depuis une source locale, vous pouvez utiliser la syntaxe `file:`. Par exemple, si les packages sont stockés sur un CD−ROM, et que ce dernier est monté sur `/mnt/cdrom`, il faut alors indiquer: `file:///mnt/cdrom`. (Noter les 3 slashes).

In which directory would you like to place the LTSP client tree
: (Dans quel répertoire souhaitez−vous installer l'arborescence LTSP pour les clients légers?) C'est le répertoire où l'arborescence utilisée par les clients légers sera créée. Par défaut, il s'agit de `/opt/ltsp`. Si ce répertoire n'existe pas, il sera créé.

  Dans ce répertoire, une arborescence sera créée pour chaque architecture de client léger. Pour l'instant, seuls les clients légers à base de x86 sont officiellement supportés par LTSP, mais plusieurs équipes travaillent sur le portage vers d'autres architectures, comme les stations PPC ou Sparc.

HTTP Proxy
: (Serveur Proxy HTTP) Si le serveur est protégé par un pare−feu, et/ou que l'accès à l'Internet passe par un serveur proxy, on peut spécifier ici l'URL du proxy à utiliser, protocole et numéro de port inclus. Par exemple: `http://firewall.mondomaine.com:3128`.

  Si aucun proxy HTTP n'est nécessaire, la valeur de ce paramètre doit être "`none`".

FTP Proxy
: (Serveur Proxy FTP) Si les packages à récupérer sont stockés sur un serveur FTP et qu'il faut passer par un proxy FTP pour y accéder, on peut en indiquer l'URL ici. La syntaxe est identique à l'option

HTTP Proxy.

Si aucun proxy FTP n'est nécessaire, la valeur de ce paramètre doit être "none".

Une fois ces paramètres confirmés, le programme d'installation va interroger le serveur de dépôt et récupérer la liste des composants disponibles.

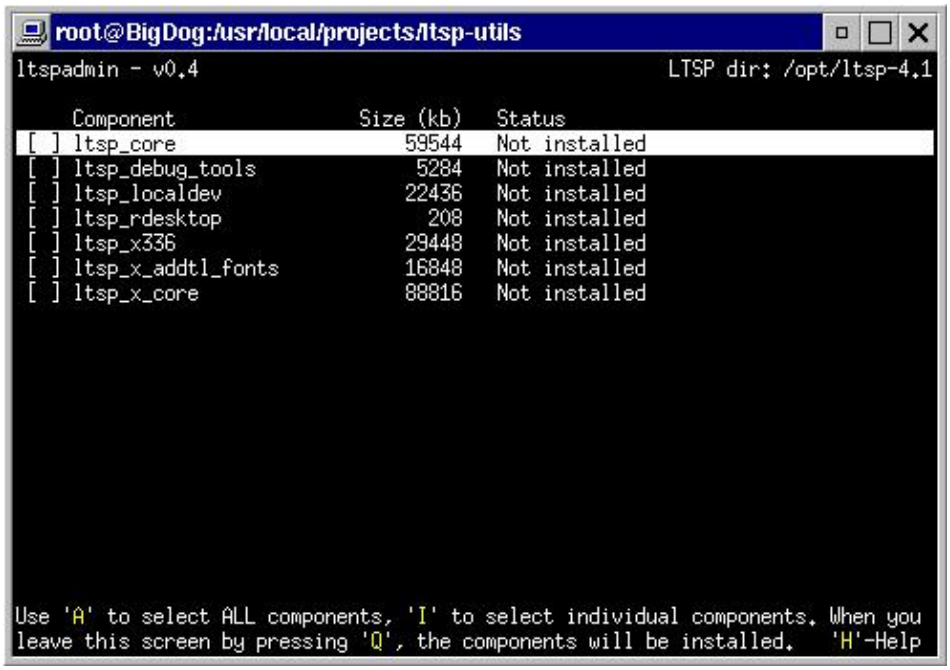

Figure 2−3. Installation de LTSP − Liste des composants

Sur cet écran, vous pouvez sélectionner les composant à récupérer et à installer. Pour ce faire, positionnez la ligne en surbrillance sur le composant désiré (à l'aide des touches de déplacement du clavier) puis appuyez sur '**I**' pour le sélectionner. Répétez l'opération pour chaque composant choisi. On peut aussi appuyer sur la touche '**A**' (A=All) pour sélectionner TOUS les composants proposés. C'est ce qui est généralement recommandé, pour supporter une plus large gamme de clients légers.

Plusieurs touches de clavier peuvent être utilisées pour naviguer sur cet écran. Pour obtenir une aide à ce propos, appuyez sur la touche '**H**' (H=Help).

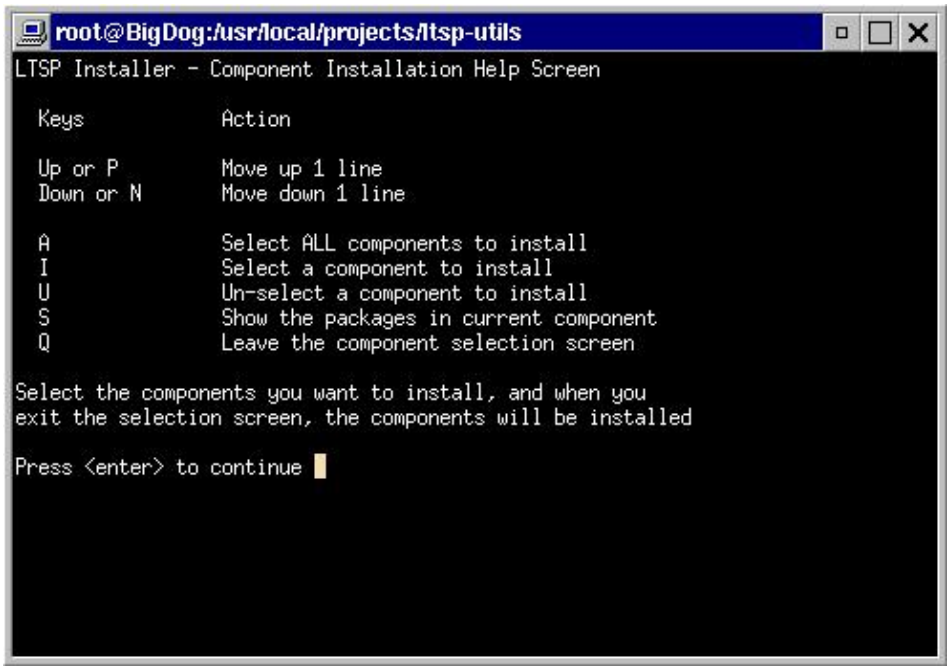

Figure 2−4. Installation de LTSP − Ecran d'aide

Par exemple, si vous souhaitez connaître la liste des packages formant un composant particulier, appuyez sur la touche '**S**' (S=Show), et la liste des packages correspondants apparaîtra, chacun avec la version courante et la dernière version disponible.

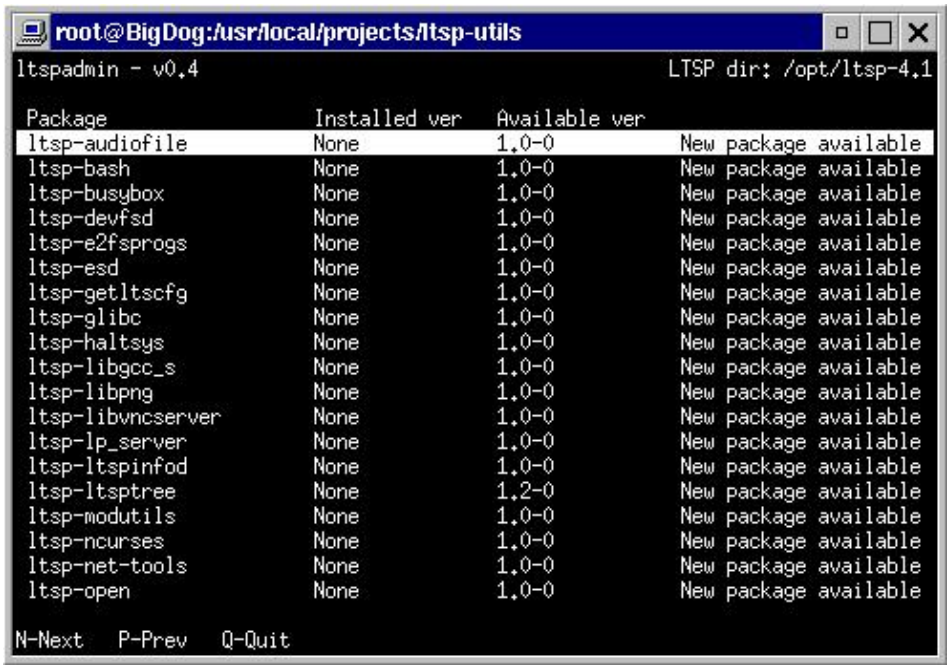

Figure 2−5. Installation de LTSP − Liste des packages d'un composant

Lorsque tous les composants désirés ont été sélectionnés, quittez l'écran de sélection avec la touche '**Q**'. Le programme demande alors si on souhaite réellement installer ou mettre à jour la sélection. Si vous répondez

'**Y**' (Y=Yes), le programme commence à télécharger les packages puis les installe sur le serveur.

2.3. Configuration des services requis par LTSP

Il y a quatre services de bases nécessaires au démarrage de clients légers avec LTSP. Ce sont :

- DHCP
- TFTP
- NFS
- XDMCP

L'utilitaire **ltspcfg** peut être utilisé pour configurer ces services, ainsi que d'autres paramètres relatifs à LTSP.

Il est possible de lancer le programme **ltspcfg** directement depuis **ltspadmin**, ou bien depuis la ligne de commande du shell, en entrant **ltspcfg**.

Lorsque **ltspcfg** est lancé, il effectue une série de contrôles sur le serveur, afin de vérifier ce qui est actuellement installé et opérationnel (activé). Un écran similaire apparaît alors:

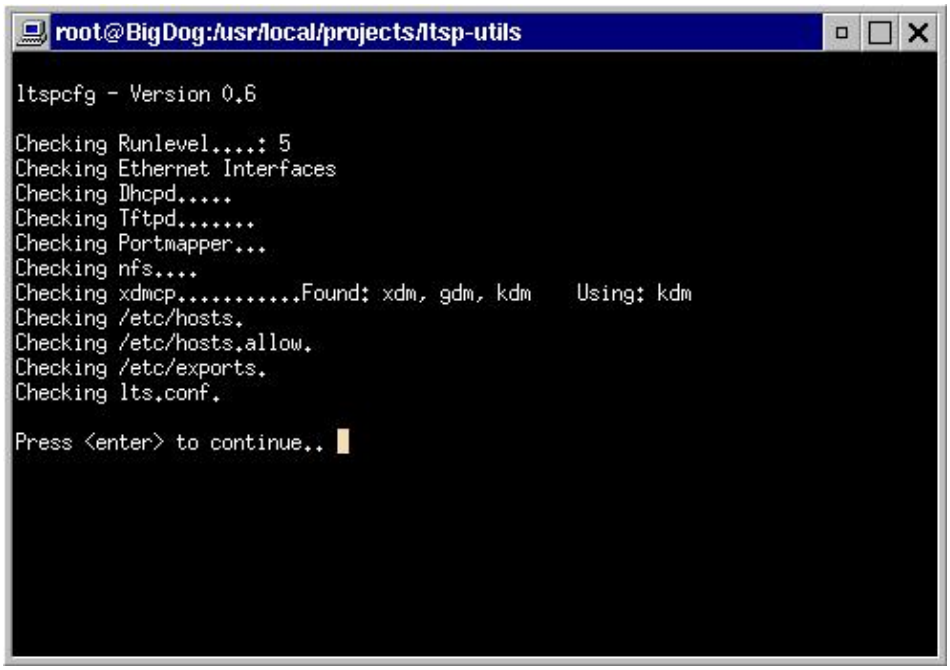

Figure 2−6. ltspcfg − Ecran initial

Cet écran présente tous les éléments contrôlés par **ltspcfg**.

Pour configurer ces éléments, appuyez sur la touche '**C**', et un écran de configuration est alors affiché. A partir de ce nouveau menu, vous pourrez vérifier chaque élément, afin de s'assurer que sa configuration est bien conforme à ce que LTSP attend.

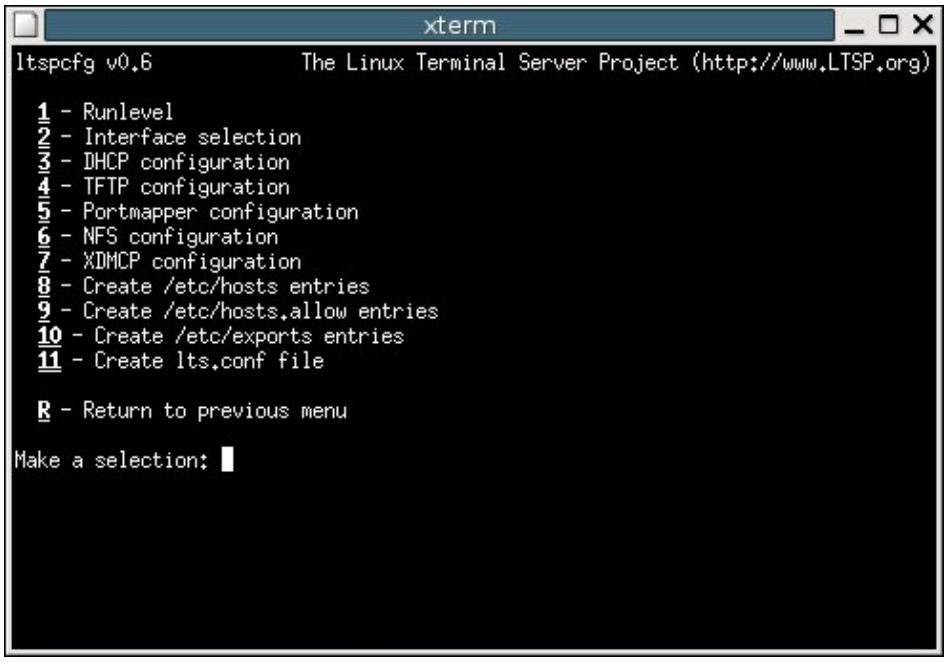

Figure 2−7. ltspcfg − Sélection des éléments à paramétrer

1 − Runlevel

Runlevel ("niveau d'exécution") est une valeur utilisée par le programme **init**. Sur les systèmes Unix et Linux, à tout moment, le système est dans un "niveau d'exécution" donné. Les niveaux 2 et 3 sont en général utilisés lorsque le serveur est en mode texte (pas de système graphique en cours). Un "runlevel" 5 indique que le serveur est passé en mode graphique avec gestion du réseau.

Pour un serveur LTSP, c'est le niveau 5 qui est utilisé. La plupart des serveurs sont déjà configurés pour que NFS et XDMCP soient activés en niveau 5. Pour les serveurs où ce n'est pas le cas, **ltspcfg** peut s'en charger.

2 − Interface selection

(Choix d'une interface réseau) Pour les serveurs équipés de plusieurs cartes réseau, il est nécessaire d'indiquer ici sur quelles interfaces seront connectés les clients légers.

Une fois le(s) interface(s) choisie(s), **ltspcfg** pourra alors créer les fichiers de configuration correspondants, comme `/etc/dhcpd.conf` et `/etc/exports`.

3 − DHCP configuration

(Configuration DHCP) DHCP doit être configuré pour fournir les informations suivantes aux clients légers en cours de démarrage: **fixed−address**, **filename**, **subnet−mask**, **broadcast−address** et **root−path**.

En sélectionnant cette option du menu, **ltspcfg** va créer le fichier `dhcpd.conf`, et activer le daemon **dhcpd** pour qu'il se lance dès le démarrage du serveur.

4 − TFTP configuration

TFTP (Trivial FTP) est un sous ensemble du protocole FTP utilisé par les clients légers pour télécharger leur noyau Linux à partir du serveur. Le daemon correspondant, **tftpd**, doit être activé sur le serveur LTSP.

5 − Portmapper configuration

Le service (daemon) **Portmapper** est utilisé par tous les programmes RPC (clients ou serveurs), comme par exemple NFS.

6 − NFS configuration

NFS est le service qui permet "d'exporter" des répertoires du serveur vers des machines distantes, où il seront montés sur l'arborescence locale. Ce service est indispensable pour les clients légers LTSP. En effet, NFS sert à monter la racine de leur système de fichier (`/`) à partir d'un répertoire du serveur (`/opt/ltsp/i386` par défaut).

Cette option du menu sert à configurer NFS, afin qu'il soit activé dès le démarrage du serveur. Le fichier de configuration `/etc/exports` est également créé (cette opération est décrite plus loin dans ce chapitre).

7 − XDMCP configuration

XDMCP signifie "X Display Manager Control Protocol". Les serveurs X (s'exécutant sur les clients légers) envoient une requête au serveur XDMCP (en général celui où est aussi installé LTSP) afin de recevoir leur écran de login.

Les gestionnaires de connexion X(X display manager) les plus courants sont **XDM**, **GDM** et **KDM**. Cette option du menu affiche les gestionnaires installés sur le serveur et celui qui est actuellement activé.

Pour des questions de sécurité, le gestionnaire de connexion X d'un serveur est souvent configuré pour ne PAS accepter la connexion d'utilisateurs depuis des machines distantes. Ce paramétrage restrictif est la cause principale des erreurs de connexion depuis les clients légers, qui affichent un premier ***écran gris avec curseur en forme de croix***, mais ne peuvent ensuite trouver un serveur XDMCP pour obtenir une connexion. Dans la plupart des cas, le programme **ltspcfg** peut configurer le gestionnaire de connexion X pour qu'il accepte les demandes de connexion en provenance des clients légers.

8 − Create `/etc/hosts` entries

(Création d'entrées dans `/etc/hosts`) Comme NFS et le gestionnaire de connexion X, de nombreux services réseau doivent trouver l'adresse IP d'un hôte (host) à partir de son nom. Il est possible d'utiliser le daemon "**bind**" (Berkeley Internet Naming Daemon) pour implémenter cette fonction (serveur de noms DNS), et il faut également s'assurer que la fonction ***inverse*** est correctement configurée (récupération du nom d'un hôte à partir de son adresse IP). Le daemon **bind** est probablement le meilleur moyen de le faire, mais le paramétrage de ce service dépasse le cadre de ce document et les possibilités de **ltspcfg**.

Une approche beaucoup plus simple de la résolution de noms est l'utilisation du fichier `/etc/hosts`.

9 − Create `/etc/hosts.allow` entries

(Création d'entrées dans `/etc/hosts.allow`) Plusieurs services réseau utilisent un système de sécurité supplémentaire appelé ***tcpwrappers***, dont la configuration se trouve dans le fichier `/etc/hosts.allow`. Cette option du menu **ltspcfg** permet de configurer ce fichier.

10 − Create the `/etc/exports` file

(Création du fichier `/etc/exports`) Ce fichier décrit les répertoires du serveur exportés par NFS et quelles sont les machines distantes autorisées à les monter dans leur arborescence locale. Cette option

du menu **ltspcfg** permet de créer ce fichier.

11 − Create the `lts.conf` file

(Création du fichier `lts.conf`) La configuration de chaque client léger devant se connecter au serveur LTSP est définie dans le fichier `lts.conf` (créé par défaut dans le répertoire `/opt/ltsp/i386/etc`). Pour des clients légers récents, équipés d'un bus PCI, il n'est pas nécessaire d'ajouter ou modifier les paramètres par défaut, mais ce fichier doit exister. Cette option du menu **ltspcfg** permet de le créer.

2.4. Configuration relative aux clients légers

Après la configuration du serveur, on peut maintenant se pencher sur les paramètres LTSP concernant les clients légers. Pour cela, trois fichiers sont utilisés.

1. `/etc/dhcpd.conf`
2. `/etc/hosts`
3. `/opt/ltsp/i386/etc/lts.conf`

2.4.1. `/etc/dhcpd.conf`

Chaque client léger devant se connecter au serveur doit obtenir une adresse IP, ainsi que d'autres données. Au démarrage (boot), le client léger va recevoir du serveur DHCP les informations suivantes:

- adresse IP ;
- nom de machine (hostname) ;
- adresse IP du serveur ;
- passerelle par défaut ;
- chemin d'accès vers le noyau à télécharger ;
- serveur et chemin d'accès vers le répertoire à monter comme racine de son système de fichier local.

Dans l'exemple qui suit, DHCP est configuré pour distribuer des adresses IP aux clients légers de son réseau.

Pendant la configuation du serveur par **ltspcfg**, un fichier `dhcpd.conf` est créé à titre d'exemple. Il se nomme `/etc/dhcpd.conf.example`. Vous pourrez ensuite le copier ou le renommer en `/etc/dhcpd.conf` et s'en servir comme base de la véritable configuration DHCP. A ce stade, il est nécessaire de modifier ce fichier pour qu'il corresponde à l'environnement réel du serveur et des clients légers.

```
default-lease-time            21600;
max-lease-time                21600;

option subnet-mask            255.255.255.0;
option broadcast-address      192.168.0.255;
option routers                192.168.0.254;
option domain-name-servers    192.168.0.254;
option domain-name            "ltsp.org";
option root-path              "192.168.0.254:/opt/ltsp/i386";

shared-network WORKSTATIONS {
subnet 192.168.0.0 netmask 255.255.255.0 {
}
}
```

```
group   {
use-host-decl-names       on;
option log-servers        192.168.0.254;

host ws001 {
        hardware ethernet     00:E0:18:E0:04:82;
        fixed-address         192.168.0.1;
        filename              "/lts/vmlinuz.ltsp";
}
}
```

Figure 2−8. /etc/dhcpd.conf

Depuis la version 2.09pre2 de LTSP, il n'est plus nécessaire de spécifier un noyau particulier pour chaque client léger. Le noyau standard supporte désormais toutes les cartes réseau reconnues par Linux. Selon les versions de LTSP, on trouvera 2 noyaux inclus dans le package ltsp−kernel. Le premier a été modifié avec le patch "Linux Progress Patch" (LPP), le second ne l'est pas. Ces noyaux sont respectivement nommés:

```
vmlinuz-2.4.9-ltsp-5
vmlinuz-2.4.9-ltsp-lpp-5
```

NDLT: les dernières versions de LTSP ne contiennent qu'un seul noyau Linux 2.4.26.

IMPORTANT: Le noyau Linux à télécharger par les clients légers est stocké dans le sous−répertoire `/tftpboot/lts`, mais le paramètre "filename" du fichier `/etc/dhcpd.conf` n'indique pas le chemin d'accès complet. La partie `/tftpboot` n'est pas spécifiée. En effet, à partir de la version 7.1 de Red Hat, le daemon TFTP est lancé avec l'option '−s'. Ceci a pour effet de demander au daemon tftpd de fonctionner en mode ***sécurisé***. En conséquence, le daemon effectue un **chroot** vers le répertoire `/tftpboot` lorsqu'il démarre. Dès ce moment, tous les fichiers accessibles par le daemon ont un chemin d'accès relatif à sa racine `/tftpboot`.

D'autres distributions Linux peuvent installer−utiliser tftpd sans l'option '−s'. Dans ce cas, il faut ajouter le préfixe `/tftpboot` au chemin d'accès vers le noyau à télécharger.

2.4.2. `/etc/hosts`

Récupération d'un nom d'hôte à partir de son adresse IP

Généralement, les ordinateurs échangent leurs adresses IP sous forme numérique (ex : 192.168.0.5). Mais pour un utilisateur, il est beaucoup plus facile de se souvenir d'un nom que d'une série de chiffres. C'est pour cela qu'on utilise le service DNS et le fichier `/etc/hosts`, permettant d'assigner un nom à une adresse IP (et vice−versa). Ce service n'est pas obligatoire, mais LTSP en a absolument besoin, en particulier lors du montage du système de fichier des clients légers via NFS. En l'absence de ce service, le système pourrait générer des erreurs d'autorisation.

Il est non seulement essentiel pour NFS, mais aussi pour les gestionnaires de connexion X ***GDM*** ou ***KDM***. Si le nom d'un client léger n'existe pas dans le fichier `/etc/hosts`, le gestionnaire de connexion ne fonctionnera pas correctement.

2.4.3. `/opt/ltsp/i386/etc/lts.conf`

La majorité des paramètres de configuration des clients légers est regroupée dans ce fichier `lts.conf`.

Le format de `lts.conf` est plutôt simple. Il est découpé en plusieurs "sections". Il y a une section générale par défaut appelée **[default]** et il peut y avoir d'autres sections, spécifiques à des clients légers ne rentrant pas dans le cas général. Le nom de ces sections spécifiques peut être le nom d'hôte du client léger, son adresse IP ou son adresse physique (MAC adress).

Voici un exemple de fichier `lts.conf` :

```
#
# Config file for the Linux Terminal Server Project (www.ltsp.org)
#

[Default]
        SERVER             = 192.168.0.254
        XSERVER            = auto
        X_MOUSE_PROTOCOL   = "PS/2"
        X_MOUSE_DEVICE     = "/dev/psaux"
        X_MOUSE_RESOLUTION = 400
        X_MOUSE_BUTTONS    = 3
        USE_XFS            = N
        LOCAL_APPS         = N
        RUNLEVEL           = 5

[ws001]
        USE_NFS_SWAP       = Y
        SWAPFILE_SIZE      = 48m
        RUNLEVEL           = 5

[ws002]
        XSERVER            = XF86_SVGA
        LOCAL_APPS         = N
        USE_NFS_SWAP       = Y
        SWAPFILE_SIZE      = 64m
        RUNLEVEL           = 3
```

Exemple 2−1. lts.conf file

Explications de certains paramètres de cet exemple :

XSERVER

Si le client léger possède une carte graphique PCI, et qu'elle est supportée par XFree86 4.1 ou Xorg, il suffit d'installer le package `lts_x_core` qui contient tous les drivers nécessaires.

LTSP contient également des packages XFree86 3.3.6 supplémentaires pour les cartes qui ne sont pas encore supportées par les serveurs XFree86 4.1 ou Xorg.

Comme pour les autres paramètres, on peut définir XSERVER dans la section [default] afin que sa valeur soit utilisée par tous les clients légers, ou dans une section spécifique à un client léger particulier.

Dans cet exemple, tous les clients légers (sauf ws002) ont une carte vidéo PCI automatiquement détectée. Il suffit de donner la valeur "auto" au paramètre XSERVER de la section [default]. Le client léger "ws002" possède une ancienne carte vidéo qui n'est reconnue que par le serveur X XF86_SVGA. On indique donc le nom de ce serveur dans la section spécifique à ce client léger.

RUNLEVEL

Pour tous les client légers (sauf ws002) qui doivent fonctionner en mode graphique, on positionnera la valeur du runlevel à "5". Et pour que le client léger ws002 fonctionne en mode caractère, on positionne le runlevel à "3" dans sa section particulière.

2.5. Affichage de la configuration courante

Le programme **ltspcfg** permet d'afficher à l'écran le status de tous les services requis par LTSP. Pour cela, appuyer sur la touche '**S**'.

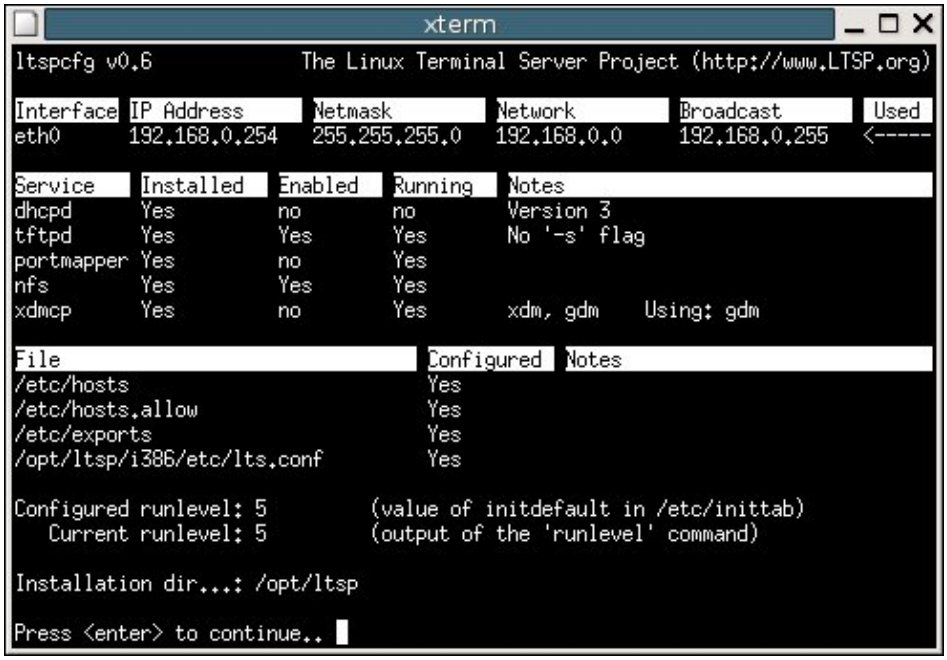

Figure 2−9. ltspcfg − Configuration courante

Chapitre 3. Paramétrage des clients légers

Le server à présent configuré, il est temps de se pencher sur la configuration des clients légers.

L'essentiel du projet LTSP concerne ce qui se passe APRES le chargement du noyau Linux en mémoire d'un client léger. Pour l'instant, nous allons étudier comment télécharger ce noyau. Il y a plusieurs façons de le faire: avec Etherboot, Netboot, PXE et depuis un lecteur de disquette.

3.1. Démarrage avec PXE

Si la carte réseau (ou le BIOS) du client léger permet l'utilisation de PXE pour démarrer, vous pouvez utiliser cette méthode pour charger le noyau Linux en mémoire. PXE est une technique de bootrom similaire à Etherboot ou Netboot.

Il peut être nécessaire d'activer préalablement PXE sur la carte réseau. De même, l'ordre dans lequel sont rangés les périphériques de démarrage doit être modifié dans le BIOS, pour que l'option "Démarrage depuis le réseau" (Boot from Lan) soit placée en première position. PXE sera alors utilisé en priorité, avant de tenter un démarrage depuis le disque dur ou le lecteur de disquette.

PXE ne peut télécharger que des fichiers dont la taille est inférieure ou égale à 32Ko. Un noyau Linux étant bien plus gros, il n'est pas possible de le charger directement avec PXE. Pour contourner ce problème, le chargement se fait en deux étapes. PXE va d'abord charger un "Network Boostrap Program" (NBP: programme de démarrage par réseau).

LTSP contient un programme NBP pour charger les noyaux Linux: **pxelinux.0**. Cet outil fait partie du package **syslinux** conçu par le développeur noyau H. Peter Anvin.

Le NBP **pxelinux.0** et son fichier de configuration se trouve dans le package **kernel** de LTSP. Il permet de télécharger Linux et son disque virtuel initial (initrd).

Le démarrage d'un client léger avec PXE se déroule de la façon suivante:

- Le bootrom PXE initialise la carte réseau et envoie une requête DHCP.
- Le serveur DHCP répond en renvoyant une adresse IP pour le client léger, et le nom du NBP à télécharger (dont la taille ne dépasse pas 32 Ko).
- Le bootrom PXE télécharge le NBP, le copie en mémoire et le lance.
- Le NBP prend le contrôle et utilise le protocole TFTP pour télécharger un fichier de configuration depuis le serveur.
- Ce fichier de configuration contient le nom du noyau, le nom du disque virtuel initial (initrd) et les options à passer au noyau une fois qu'il sera chargé.

  Voici un exemple de fichier de configuration pour pxelinux :

  ```
  prompt=0
  label linux
          kernel bzImage-2.4.24-ltsp-4
          append init=/linuxrc rw root=/dev/ram0 initrd=initrd-2.4.24-ltsp-4.gz
  ```

- Le NBP utilise à nouveau TFTP pour télécharger le noyau Linux indiqué, ainsi que son disque virtuel initial (initrd).

- Lorsque le noyau est complètement chargé en mémoire, le NBP lui passe le contrôle Le noyau démarre en montant son disque virtuel, puis continue son initialisation comme cela est détaillé dans le chapitre "Principes de fonctionnement".

3.2. Démarrage avec Etherboot

> Etherboot est un package logiciel destiné à créer des images de BOOT ROM. Ces BOOT ROM peuvent télécharger des programmes sur un réseau Ethernet, pour des ordinateurs de type x86. De nombreuses cartes réseau ont un emplacement (socket) libre destiné à accueillir une BOOT ROM. C'est dans cette BOOT ROM que l'on copie le code Etherboot.
>
> −−Ken Yap

Etherboot est un logiciel libre, distribué sous licence "GNU General Public License, Version 2 (GPL2)".

Pour utiliser Etherboot, si le client léger possède une carte réseau avec une BOOT ROM Etherboot, il faudra sans doute paramétrer le BIOS pour que ce dernier démarre en priorité depuis la carte réseau (Boot from LAN) avant d'essayer depuis le disque dur, le lecteur de disquette ou le CD−ROM.

Si vous ne possèdez pas de BOOT ROM, vous pouvez en programmer ("brûler") une avec un appareil prévu à cet effet avant de l'installer sur la carte réseau. Vous pouvez également copier le code Etherboot sur le secteur de démarrage d'une disquette et démarrer à partir du lecteur de disquette.

Etherboot gère une vaste gamme de cartes réseau, plus de 200 modèles, et cette liste s'enrichit régulièrement. Qu'il s'agisse de brûler le code Etherboot dans une BOOT ROM (Eprom) ou de le copier sur une disquette, la seule chose à connaître est le type précis de la carte réseau du client léger.

3.2.1. Sélection d'un driver Etherboot pour une carte réseau ISA

Pour les anciennes cartes ISA, il n'est pas indispensable de connaître son type exact. La plupart d'entre elles sont de type ne2000 ou 3com 3c509. La seule chose qui compte est de choisir le driver en fonction du câblage 10 base−T (Coaxial) ou 10 base−2 (paire torsadée).

3.2.2. Sélection d'un driver pour une carte réseau PCI

Pour les cartes PCI, il faut par contre choisir le driver correspondant exactement à l'identifiant PCI de la carte et de son fabricant.

La plupart du temps, on peut facilement récupérer ces informations, car elles sont imprimées sur la carte et correspondent à la description du driver. Mais il arrive qu'on ne puisse pas les retrouver.

Dans ce cas, si le client léger est équipé d'un lecteur de disquette, on peut y insérer une disquette tomsrtbt (Tom's Root Boot). Ou bien, si le client léger dispose d'un lecteur CD−ROM, on peut démarrer avec un CD Linux, comme Knoppix. S'il n'est pas possible de lancer Linux sur le client léger pour découvrir le type de le carte réseau, le seul moyen restant est d'installer la carte réseau sur une machine où cela est possible. Linux.

Lorsque Linux est chargé, vous pouvez alors utiliser la commande **lspci** avec l'option '−n'.

```
[root@jamlap root]# lspci −n
0000:00:00.0 Class 0600: 8086:7190 (rev 03)
```

```
0000:00:01.0 Class 0604: 8086:7191 (rev 03)
0000:00:03.0 Class 0607: 104c:ac1c (rev 01)
0000:00:03.1 Class 0607: 104c:ac1c (rev 01)
0000:00:07.0 Class 0680: 8086:7110 (rev 02)
0000:00:07.1 Class 0101: 8086:7111 (rev 01)
0000:00:07.2 Class 0c03: 8086:7112 (rev 01)
0000:00:07.3 Class 0680: 8086:7113 (rev 03)
0000:00:08.0 Class 0401: 125d:1978 (rev 10)
0000:01:00.0 Class 0300: 1002:4c4d (rev 64)
0000:06:00.0 Class 0200: 8086:1229 (rev 09)
```

Dans l'exemple ci−dessus, **lspci** affiche une ligne pour chaque carte PCI installée dans la machine. La seule ligne qui compte concerne les composants de classe ***Class 0200***. On peut donc filtrer le résultat de **lspci** pour obtenir quelque chose de plus simple.

```
[root@jamlap root]# lspci −n | grep "Class 0200"
0000:06:00.0 Class 0200: 8086:1229 (rev 09)
```

Les identifiants PCI recherchés sont **8086:1229**. Le premier champ, **8086** correspond à l'identifiant du fabricant. Dans cet exemple, il s'agit d'Intel Corporation. Le deuxième champ, **1229** est l'identifiant de la carte, et il s'agit ici d'une EtherExpress 100 card. NDLT : Voir aussi **lspci** avec l'option '−v'.

3.2.3. Création d'une disquette de démarrage

Pour créer un bootrom Etherboot, il faut d'abord télécharger le package Etherboot, puis l'installer et le configurer pour le type de bootrom souhaité. Ensuite on compilera le source pour générer le code Etherboot. Ce dernier peut alors être brûlé dans une BOOT ROM (EPROM), ou copié sur disquette.

Mais il existe un moyen beaucoup plus simple d'arriver au même résultat, grâce au site Internet de Marty Connor *www.Rom−O−Matic.net*.

Marty a réalisé un excellent travail, en créant une interface permettant de créer un bootrom Etherboot à partir d'un simple navigateur web. Toutes les opérations de configuration et de compilation sont effectuées sur son serveur. Il suffit de choisir le type de carte et de bootrom dans les listes proposées, et d'indiquer quelques autres paramètres. Il ne reste plus ensuite qu'à appuyer sur le bouton 'Get ROM' pour qu'un bootrom soit généré !

Pour pouvoir copier le bootrom sur une disquette, choisissez l'option 'Floppy Bootable ROM Image'. Ceci va insérer un en−tête de 512 octets dans dans le bootrom. Lorsque le client léger démarre, cet en−tête charge le bootrom en mémoire, où il est exécuté.

Appuyez sur le bouton 'Get ROM'. Après quelques secondes, une nouvelle fenêtre apparaît dans le navigateur, demandant où sauvegarder le fichier bootrom. Il n'y a plus qu'à choisir un emplacement sur le disque dur local, et le bootrom est aussitôt téléchargé.

Une fois sauvegardé sur le disque dur, il faut ensuite copier le bootrom sur une disquette. Insérez une disquette dans le lecteur et lancez la commander suivante:

```
dd if=Etherboot_Image of=/dev/fd0
```

3.2.4. Création d'un bootrom

Pour écrire (on dit aussi "brûler") un bootrom Etherboot dans une EPROM, il faut un programmateur d'EPROM. On trouve ce type d'appareil dans toutes les gammes, de quelques centaines d'euros à plusieurs milliers, selon les fonctionnalités offertes.

La copie d'un bootrom dans une EPROM varie d'un programmateur à l'autre. La description de cette opération dépasse le cadre de ce document.

Chapitre 4. Démarrage du client léger

En assumant que le serveur et le client léger ont été correctement configurés, il ne reste plus qu'à insérer la disquette dans le lecteur et à mettre le client léger sous tension.

Le code Etherboot est alors lu depuis la disquette et chargé en mémoire, la carte réseau est identifiée et initialisée, une requête DHCP est envoyée au serveur, qui répond à son tour, puis le noyau Linux est téléchargé dans la mémoire du client léger. Une fois que le noyau aura initialisé le matériel du client léger, le serveur X Window démarrera et une fenêtre d'identification (login) devrait s'afficher à l'écran, comme dans l'exemple ci dessous.

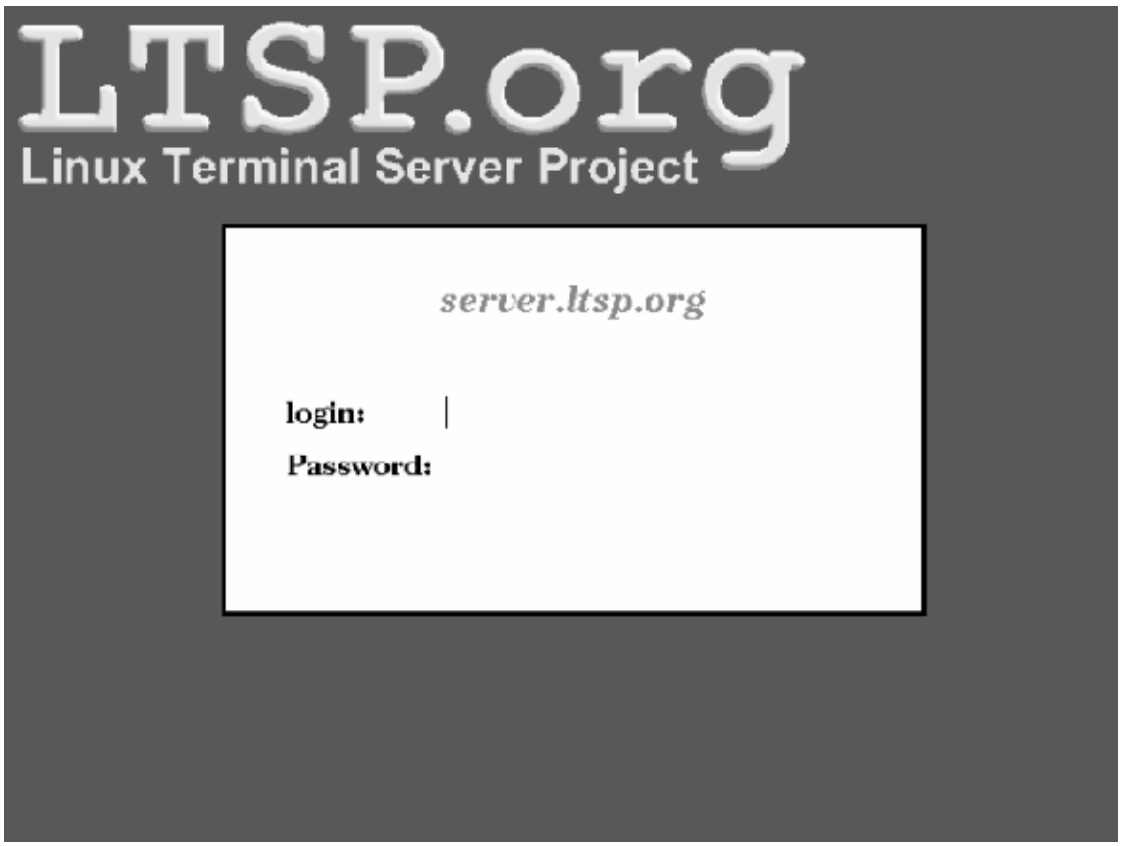

Figure 4−1. Login screen

A partir de là, il est possible de s'identifier et de se connecter au serveur. Il faut se rappeler que toutes les commandes lancées sur le client léger sont en fait exécutées sur le serveur. Il n'y a que l'affichage qui est déporté sur le client léger. C'est un des points forts de la technologie X Window.

Vous pouvez maintenant utiliser n'importe quel programme installé sur le serveur.

Chapitre 5. Editions et imprimantes

En dehors de son utilisation comme terminal graphique ou caractère, un client léger peut aussi fonctionner (simultanément) comme un serveur d'impression. Jusqu'à 3 imprimantes peuvent être rattachées sur les ports parallèles et série du client léger.

Ceci est totalement transparent pour l'utilisateur du client léger. Le trafic supplémentaire qui passe au travers du client léger vers les imprimantes n'a pas ou peu d'impact sur son fonctionnement.

5.1. Configuration côté client léger

Sur le client léger, le programme **lp_server** redirige les travaux d'impression en provenance du serveur vers les imprimantes locales.

Pour activer une imprimante sur un client léger, il faut l'indiquer dans le fichier de configuration du serveur: `lts.conf`.

```
[ws001]
PRINTER_0_DEVICE = /dev/lp0
PRINTER_0_TYPE   = P
```

Ce paramètre va demander au programme **lp_server** de fonctionner en tant que daemon sur le client léger, à l'écoute sur le port TCP/IP 9100. Ce port recevra le flux d'impression en provenance du serveur. Le daemon **lp_server** va rediriger ce flux vers l'imprimante connectée au port parallèle `/dev/lp0` du client léger.

Il existe de nombreux autres paramètres d'impression. Voir le chapitre relatif au fichier `lts.conf` pour plus de détails.

5.2. Configuration côté serveur

Côté serveur, la configuration consiste à définir une file d'impression à l'aide de l'outil graphique de gestion des imprimantes.

Sur un serveur Redhat 7.2, on dispose d'un outil de configuration en mode caractère et en mode graphique. La version graphique se nomme **printconf−gui**, et la versions texte **printconf−tui**. Les versions plus anciennes contiennent un programme nommé **printtool**. Les autres distributions Linux disposent de leurs propres outils de configuration (comme CUPS).

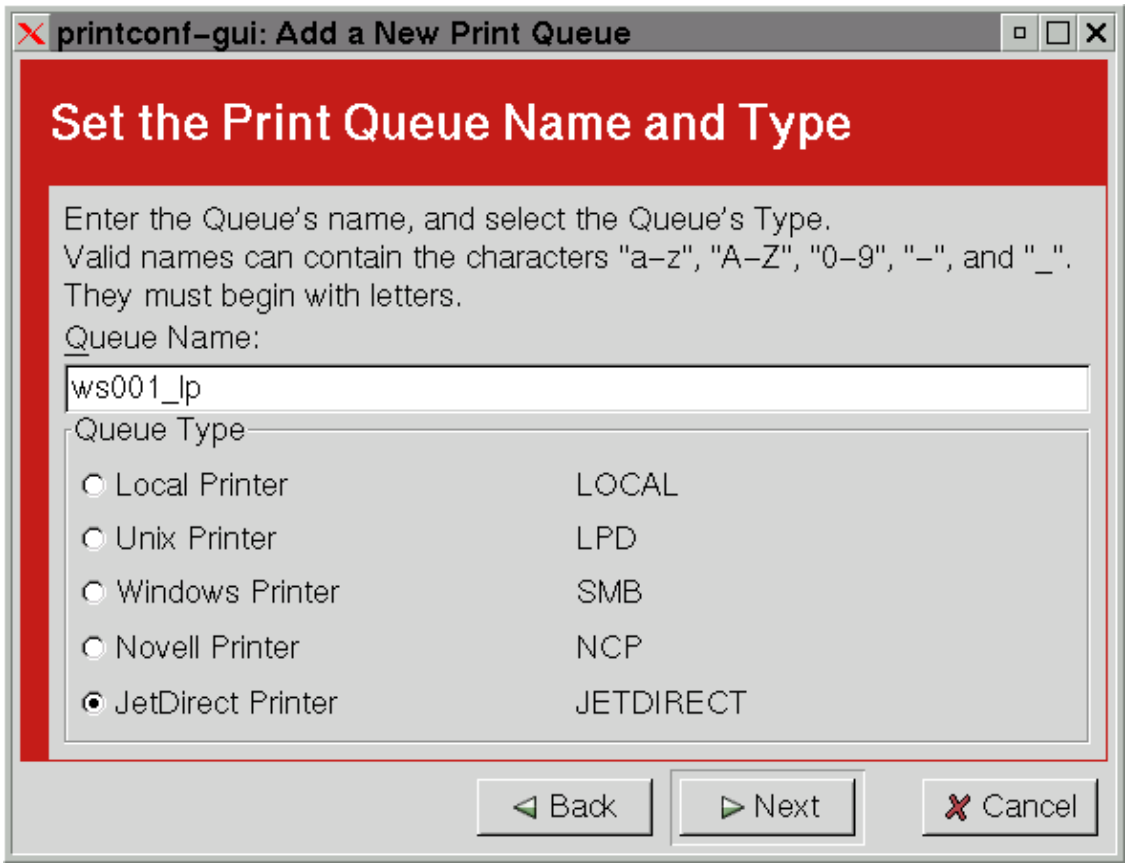

Figure 5−1. Ajout d'une imprimante avec Printconf−gui

Une fois le programme lancé, il faut créer une nouvelle imprimante. Dans cet exemple, on utilise le mode émulation HP JetDirect supporté par **lp_server**. Il suffit donc de créer une imprimante **JetDirect**.

Pour accéder à cette imprimante, il faut définir un nom de file d'impression (queue). N'importe quel nom peut être spécifié, mais autant donner un nom significatif. Les caractères acceptés dans un nom sont :

- `"a-z" les lettres minuscules`
- `"A-Z" les majuscules`
- `"0-9" les chiffres`
- `"-"   le tiret`
- `"_"   le tiret bas (souligné)`

Le nom choisi dans cet exemple est **ws001_lp**, ce qui permet de comprendre que l'imprimante concernée est connectée au client léger **ws001**.

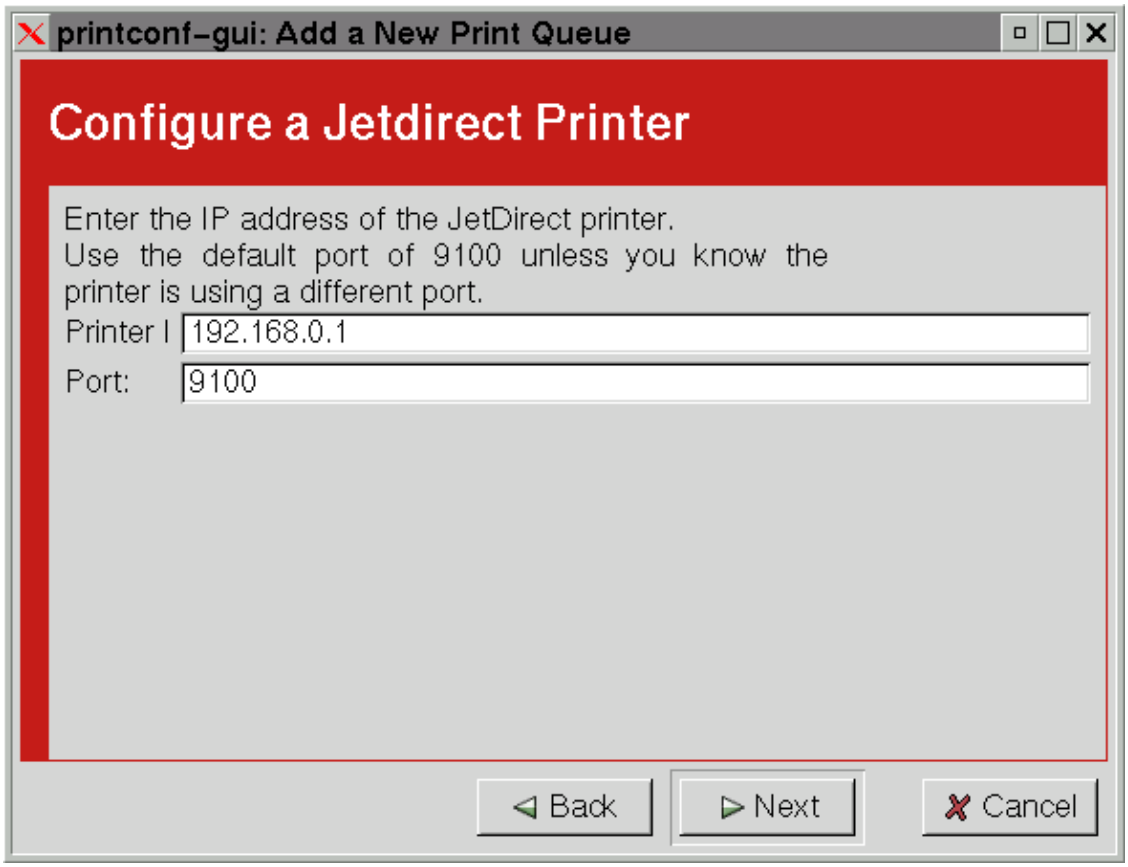

Figure 5−2. Paramétrage de l'imprimante avec Printconf−gui

Pour communiquer avec l'imprimante, deux autres champs doivent être renseignés :

1. Adresse IP ou nom de l'hôte sur lequel est connectée l'imprimante.
2. Numéro de port sur lequel le daemon **lp_server** est à l'écoute (en attente).

   Pour la première imprimante connectée à un client léger, on utilise le port TCP/IP **9100**. Pour la deuxième, on utilisera le port **9101**, et **9102** pour la troisième.

Chapitre 6. Scripts d'écrans

Depuis la version 4.0 de LTSP, une nouvelle fonctionnalité appelée **Screen Scripts** (Scripts d'écrans) est disponible. Les scripts permettent de lancer différents types de sessions sur les clients légers.

Il est possible de définir plusieurs scripts d'écrans pour un même client léger. Ils peuvent être de types différents ou du même type. On peut par exemple spécifier :

```
SCREEN_01 = startx
SCREEN_02 = shell
```

Ceci aura pour effet de lancer une session X sur le premier écran, et un shell local sur le second. Pour passer d'un écran à l'autre, utilisez la combinaison de touches Ctrl−Alt−Fn, où 'n' est un numéro de 1 à 12. Ctrl−Alt−F1 bascule vers le premier écran, Ctrl−Alt−F2 vers le deuxième, etc.

Même s'il est possible de créer jusqu'à 12 écrans différents, on n'en utilise en général qu'un seul.

Les types d'écran disponibles sont les suivants:

startx

Ce script d'écran lance sur le client léger un serveur X avec l'option −query, ce qui a pour effet d'envoyer une requête XDMCP à un gestionnaire de connexion X à l'écoute sur le réseau. Le gestionnaire qui répond renvoie au client léger un écran d'identification (login).

shell

Ce script d'écran lance un shell sur l'écran du client léger. Il s'agit d'une session LOCALE sur le client léger, ce qui offre peu d'interêt, sauf pour la mise au point en cas de problème. A partir de ce shell, il n'est pas possible d'exécuter des applications se trouvant sur le serveur.

telnet

Ce script d'écran lance sur le client léger une session telnet pour une connexion vers le serveur. Cette session ouvre une session en mode caractère sur le serveur.

Par défaut, telnet tente de se connecter sur le serveur LTSP. Il est possible de spécifier un autre serveur, en ajoutant un paramètre supplémentaire au script d'écran. Exemple :

```
SCREEN_01 = telnet server2.mydomain.com
```

On peut également ajouter n'importe quelle option acceptée par telnet. Mais dans ce cas, il faut aussi spécifier le nom du serveur à contacter.

rdesktop

Ce script d'écran lance le programme **rdesktop**, qui établit une connexion sur un serveur Microsoft Windows. On peut spécifier en option n'importe quel paramètre reconnu par **rdesktop**, directement après le nom du script. Par exemple, pour indiquer le nom du serveur à contacter :

```
SCREEN_01 = rdesktop −f w2k.mydomain.com
```

Ceci va lancer **rdesktop** en mode plein écran. L'utilisateur voit apparaître un écran de logon Windows, à partir duquel il peut s'identifier et se connecter. Une fois connecté, l'utilisateur ne s'aperçoit même pas que son client léger fonctionne sous Linux.

Ces scripts d'écrans sont stockés dans le répertoire `/opt/ltsp/i386/etc/screen.d`. Vous pouvez créer d'autres scripts et les placer dans le même répertoire pour qu'ils soient pris en compte. Il est conseillé de s'inspirer des scripts existants pour en créer de nouveaux.

Chapitre 7. Diagnostic et mise au point

Après avoir suivi les instructions des chapitres précédents, si un client léger ne démarre pas de la manière attendue, vous devrez alors en rechercher les causes et tenter de corriger le problème. Ce chapitre décrit les principaux problèmes rencontrés.

La première chose à faire est de vérifier à quelle étape le processus de démarrage a rencontré un problème.

7.1. Démarrage Etherboot

Lorsqu'on démarre un client léger à partir d'une disquette Etherboot, l'écran doit en principe afficher un message similaire à ce qui suit:

```
loaded ROM segment 0x0800 length 0x4000 reloc 0x9400
Etherboot 5.0.1 (GPL) Tagged ELF for [LANCE/PCI]
Found AMD Lance/PCI at 0x1000, ROM address 0x0000
Probing...[LANCE/PCI] PCnet/PCI−II 79C970A base 0x1000, addr 00:50:56:81:00:01
Searching for server (DHCP)...
<sleep>
```

Cet exemple montre ce qui doit normalement se passer lorsqu'on démarre depuis une disquette Etherboot. Si ces messages n'apparaissent pas, il y a de grandes chances que la disquette soit défectueuse, ou que le code Etherboot n'a pas été copié correctement.

Si un message similaire à ce qui suit apparaît, il se peut alors que le code Etherboot utilisé ne corresponde pas à la carte réseau installée sur le client léger.

```
ROM segment 0x0800 length 0x8000 reloc 0x9400
Etherboot 5.0.2 (GPL) Tagged ELF for [Tulip]
Probing...[Tulip]No adapter found
<sleep>
<abort>
```

Si la carte est détectée et que son adresse MAC est affichée, la disquette et le code Etherboot qu'elle contient sont corrects.

7.2. DHCP

Lorsque la carte réseau est correctement initialisée, elle envoie un broadcast DHCP sur le réseau local, afin de contacter un serveur DHCP.

Si le client léger reçoit une réponse valide d'un serveur DHCP, la carte réseau est correctement configurée. On peut le vérifier lorsque l' adresse IP renvoyée par le serveur apparaît à l'écran. En voilà un exemple :

```
ROM segment 0x0800 length 0x4000 reloc 0x9400
Etherboot 5.0.1 (GPL) Tagged ELF for [LANCE/PCI]
Found AMD Lance/PCI at 0x1000, ROM address 0x0000
Probing...[LANCE/PCI] PCnet/PCI−II 79C970A base 0x1000, addr 00:50:56:81:00:01
Searching for server (DHCP)...
<sleep>
Me: 192.168.0.1, Server: 192.168.0.254, Gateway 192.168.0.254
```

Si la ligne commençant par 'Me:' suivi d'une adresse IP est affichée, on peut considérer que DHCP fonctionne correctement. On peut passer au diagnostic suivant pour vérifier TFTP.

Si, au contraire, on voir apparaître le message ci−dessous, suivi par plusieurs messages <sleep>, quelque chose ne va pas au niveau DHCP. Notez toutefois qu'il est courant de voir un ou deux messages <sleep> avant que le serveur DHCP ne réponde.

```
Searching for server (DHCP)...
```

Diagnostiquer ce qui se passe exactement est parfois difficile, mais on peut déjà explorer les pistes suivantes.

7.2.1. Vérifier les connexions et le câblage

Est−ce que le client léger est physiquement relié au même réseau que le serveur ?

Lorsque le client léger est sous tension, vérifier que le LED "Link" de la carte réseau est allumé, ainsi que sur l'appareil (switch, hub, routeur) à l'autre bout du câble.

S'il s'agit d'une connexion directe entre un client léger et un serveur (sans hub ou switch), il faut s'assurer que le câble est un câble croisé. Si la connexion passe par un hub ou un switch, il faut alors utiliser des câbles droits.

7.2.2. Est−ce que DHCP est activé ?

Il se peut que le daemon **dhcpd** ne soit pas en cours d'exécution sur le serveur. On peut le vérifer de plusieurs façons.

dhcpd s'exécute en tâche de fond (background), à l'écoute sur le port UDP 67. La commande **netstat** permet de vérifier si un processus est à l'écoute sur ce port :

```
netstat −an | grep ":67 "
```

Un message similaire à ce qui suit devrait apparaître :

```
udp     0    0   0.0.0.0:67          0.0.0.0:*
```

La 4ème colonne indique l'adresse IP et le port, séparés par deux points (':'). Une adresse composée uniquement de zéros ('0.0.0.0') indique une écoute du port 67 sur toutes les interfaces réseau. C'est le cas si un serveur possède par exemple une carte **eth0** et une autre **eth1** et que **dhcpd** est à l'écoute sur ces deux interfaces.

Ce n'est pas parce que netstat montre qu'un processus est à l'écoute sur le port UDP 67 qu'il s'agit pour autant du daemon **dhcpd**. Cela peut être un autre programme, comme **bootpd**, bien que cela soit peu probable, car le daemon **bootp** n'est plus fourni par défaut dans les distributions Linux récentes.

Pour être absolument sûr que c'est bien **dhcpd** qui fonctionne, on peut s'en assurer avec la commande **ps**.

```
ps aux | grep dhcpd
```

Une ligne équivalente à celle ci−dessous doit alors s'afficher :

```
root 23814 0.0 0.3 1676 820 ?      S 15:13 0:00 /usr/sbin/dhcpd
root 23834 0.0 0.2 1552 600 pts/0  S 15:52 0:00 grep dhcp
```

La première ligne prouve effectivement que **dhcpd** est en cours d'exécution. La deuxième correspond à l'exécution de la commande **grep**.

S'il n'y a aucune ligne montrant que **dhcpd** est lancé, il faut alors vérifier que le serveur est configuré pour fonctionner en runlevel (niveau d'exécution) 5, et que **dhcpd** est lui−même paramétré pour démarrer en

runlevel 5. Sur un serveur Redhat, la commande **ntsysv** permet de le vérifier.

On peut aussi tenter de lancer manuellement le daemon **dhcpd** avec la commande suivante :

```
service dhcpd start
```

Vérifiez les messages affichés, car le démarrage peut échouer.

7.2.3. Re−vérifier la configuration de dhcpd

Est−ce que le fichier `/etc/dhcpd.conf` contient une section pour le client léger qui ne démarre pas ?

Si on a opté pour une adresse IP fixe (fixed−address) pour ce client léger, re−vérifiez que l'adresse MAC spécifiée correspond bien à celle de sa carte réseau.

7.2.4. Est−ce que ipchains et/ou iptables bloquent la requête ?

7.2.4.1. Vérification d'ipchains

Lancez la commande suivante:

```
ipchains −L −v
```

Si ce programme est installé et qu'un message équivalent apparaît:

```
Chain input (policy ACCEPT: 229714 packets, 115477216 bytes):
Chain forward (policy ACCEPT: 10 packets, 1794 bytes):
Chain output (policy ACCEPT: 188978 packets, 66087385 bytes):
```

le problème n'est pas lié à ipchains (pas de filtrage).

7.2.4.2. Vérification d'iptables

Lancer la commande suivante:

```
iptables −L −v
```

Si ce programme est installé et qu'un message équivalent apparaît:

```
Chain INPUT (policy ACCEPT 18148 packets, 2623K bytes)
pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 17721 packets, 2732K bytes)
pkts bytes target     prot opt in     out     source               destination
```

Ce n'est pas non plus iptables qui pose problème.

7.2.5. Est−ce que le client léger envoie bien une requête ?

Vérifiez le contenu du journal `/var/log/messages` pendant que le client léger démarre. On peut le faire avec la commande suivante:

```
tail −f /var/log/messages
```

Ceci affiche le journal au fur et à mesure que des lignes y sont ajoutées.

```
server dhcpd: DHCPDISCOVER from 00:50:56:81:00:01 via eth0
server dhcpd: no free leases on subnet WORKSTATIONS
server dhcpd: DHCPDISCOVER from 00:50:56:81:00:01 via eth0
server dhcpd: no free leases on subnet WORKSTATIONS
```

Si des messages 'no free leases' apparaissent, cela signifie que **dhcpd** fonctionne bien, mais qu'il n'a aucune information lui permettant d'identifier le client léger qui demande une adresse IP.

7.3. TFTP

Etherboot utilise le protocole TFTP pour télécharger le noyau Linux depuis le serveur. C'est un protocole relativement simple, mais un mauvais paramétrage peut l'empêcher de fonctionner.

Si un message équivalent à ce qui suit apparaît sur l'écran du client léger :

```
Loading 192.168.0.254:/lts/vmlinuz−2.4.24−ltsp−4.........
```

suivi d'une série de points remplissant rapidement l'écran, alors TFTP fonctionne correctement et le noyau Linux est en cours de téléchargement.

Si, au contraire, on ne voit pas les points s'afficher, le problème est peut−être l'un des suivants.

7.3.1. tftpd n'est pas lancé

Si **tftpd** n'est pas activé, il ne répondra pas aux requêtes du client léger. Pour vérifier qu'il est bien actif, utilisez la commande **netstat** comme ceci:

```
[root@bigdog]# netstat −anp | grep ":69 "

udp     0   0 0.0.0.0:69         0.0.0.0:*                453/inetd
```

Si aucun message n'apparaît, le daemon **tftpd** n'est pas lancé.

Il y a généralement deux méthodes pour lancer **tftpd** automatiquement, soit avec **inetd**, soit avec **xinetd**, qui est plus récent.

inetd est contrôlé par le fichier de configuration `/etc/inetd.conf`. Dans ce fichier, il faut s'assurer que la ligne demandant le démarrage du daemon tftp n'est PAS mise en commentaire. Normalement la ligne doit correspondre à ceci :

```
tftp dgram udp wait nobody /usr/sbin/tcpd  /usr/sbin/in.tftpd −s /tftpboot
```

xinetd utilise un répertoire contenant autant de fichiers de configuration que de services contrôlés par xinetd. Si le serveur est équipe de xinetd, le fichier de configuration pour tftpd est `/etc/xinetd.d/tftp`. En voici un exemple de contenu:

```
service tftp
{
disable          = no
socket_type      = dgram
protocol         = udp
wait             = yes
user             = root
```

```
server          = /usr/sbin/in.tftpd
server_args     = −s /tftpboot
}
```

S'assurer que le paramètre **disable** a BIEN la valeur **no**.

7.3.2. Le noyau ne se trouve pas là où tftpd le cherche

Le noyau Linux pour les clients légers doit être stocké dans un répertoire auquel le daemon tftpd peut accéder. Si l'option '−s' est utilisée pour lancer le daemon **tftpd**, le chemin d'accès au noyau demandé par le client léger doit être relatif à `/tftpboot`. Donc, si le paramètre **filename** du fichier `/etc/dhcpd.conf` a pour valeur `/lts/vmlinuz−2.4.24−ltsp−4`, le véritable chemin d'accès au noyau doit être `/tftpboot/lts/vmlinuz−2.4.24−ltsp−4`

7.4. Montage du système de fichiers via NFS

Plusieurs motifs peuvent empêcher le montage de la racine (`/`) du système de fichier d'un client léger, et notamment:

7.4.1. No init found

Aorès le chargement du noyau, si l'écran du client léger affiche un message du type:

```
Kernel panic: No init found.  Try passing init= option to kernel.
```

il est fort probable que le répertoire monté à la racine ne soit pas le bon, ou que le répertoire `/opt/ltsp/i386` soit vide.

7.4.2. Server returned error −13

Si l'erreur suivant apparaît:

```
Root−NFS: Server returned error −13 while mounting /opt/ltsp/i386
```

ceci indique que le répertoire `/opt/ltsp/i386` n'est pas référencé dans le fichier `/etc/exports` utilisé par NFS.

Vérifier également le journal `/var/log/messages`. Un ligne du type:

```
Jul 20 00:28:39 bigdog rpc.mountd: refused mount request from ws004
                for /opt/ltsp/i386 (/): no export entry
```

confirmera le diagnostic: le contenu du fichier `/etc/exports` est incorrect.

7.4.3. Problèmes liés au daemon NFS Daemon (portmap, nfsd & mountd)

Les problèmes NFS peuvent être complexes à résoudre. Comprendre comment il faut le paramétrer, et connaître les outils disponibles pour un bon diagnostic sont autant de moyens pour trouver la solution.

Trois daemons sont nécessaires pour que NFS fonctionne correctement: **portmap**, **nfsd** et **mountd**.

7.4.3.1. Le Portmapper (portmap)

Si le client léger affiche des messages du type:

```
Looking up port of RPC 100003/2 on 192.168.0.254
portmap: server 192.168.0.254 not responding, timed out
Root−NFS: Unable to get nfsd port number from server, using default
Looking up port of RPC 100005/2 on 192.168.0.254
portmap: server 192.168.0.254 not responding, timed out
Root−NFS: Unable to get mountd port number from server, using default
mount: server 192.168.0.254 not responding, timed out
Root−NFS: Server returned error −5 while mounting /opt/ltsp/i386
VFS: unable to mount root fs via NFS, trying floppy.
VFS: Cannot open root device "nfs" or 02:00
Please append a correct "root=" boot option
Kernel panic: VFS: Unable to mount root fs on 02:00
```

il est plus que probable que le daemon **portmap** n'est pas lancé sur le serveur. Vérifiez avec le programme **ps**:

```
ps −e | grep portmap
```

Si portmap est bien lancé, le résultat doit être du type:

```
30455 ?        00:00:00 portmap
```

Essayez aussi avec la commande **netstat**. Puisque le daemon portmap utilise les ports TCP et UDP 111, la commande:

```
netstat −an | grep ":111 "
```

doit renvoyer un résultat du type:

```
tcp   0   0 0.0.0.0:111       0.0.0.0:*          LISTEN
udp   0   0 0.0.0.0:111       0.0.0.0:*
```

Si ce n'est pas le cas, **portmap** n'est pas lancé sur le serveur. Pour le lancer manuellement, utiliser la commande:

```
/etc/rc.d/init.d/portmap   start
```

Puis, assurez−vous qu'il est configuré pour se lancer automatiquement lorsque le serveur (re)démarre, avec la commande **ntsysv**

7.4.3.2. Les daemons NFS et MOUNT (nfsd & mountd)

Les deux autres dameons de NFS sont **nfsd** et **mountd**. Ils sont tous les deux lancés par le script `/etc/rc.d/init.d/nfs`.

On verifie qu'ils sont bien lancés avec la commande **ps**:

```
ps −e | grep nfs
ps −e | grep mountd
```

Si une de ces deux commandes ne renvoie aucun message, le daemon correspondant ne fonctionne pas, et il faut alors le lancer manuellement.

En principe, il est possible de les relancer tous les deux en utilisant l'argument **restart** avec le script `/etc/rc.d/init.d/nfs`, mais il semble que ce script (selon les versions) ne relance pas correctement le daemon **nfsd**. Il lance seulement **mountd** (bug?). Donc, pour être absolument sûr de (re)lancer les deux dameons, utilisez les commandes suivantes:

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

La fonction **stop** peut renvoyer une erreur, mais on peut la négliger. Par contre, la fonction **start** doit absolument renvoyer le status **OK**.

Si les deux daemons sont lancés et que NFS ne fonctionne toujours pas, vérifiez qu'ils se sont bien enregistrés auprès du portmapper. Utiliser la commande **rpcinfo**:

```
rpcinfo −p localhost
```

Le résultat doit au moins contenir les lignes suivantes:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100021    1   udp  32771  nlockmgr
100021    3   udp  32771  nlockmgr
100021    4   udp  32771  nlockmgr
100005    1   udp    648  mountd
100005    1   tcp    651  mountd
100005    2   udp    648  mountd
100005    2   tcp    651  mountd
100005    3   udp    648  mountd
100005    3   tcp    651  mountd
100024    1   udp    750  status
100024    1   tcp    753  status
```

Ceci indique que les daemons **nfs** (nfsd) et **mountd** sont lancés et bien enregistrés auprès du portmapper.

7.5. Serveur X

Les problèmes liés au serveur X d'un client léger LTSP sont sans aucun doute les plus difficiles à corriger. Si le client léger est équipé d'une carte vidéo récente, supportée par Xorg ou XFree86 v4.1, et que l'écran utilisé supporte une large gamme de fréquences et de résolutions, résoudre le problème est assez facile. Dans ce cas, l'erreur la plus probable est que le serveur X choisi n'est pas le bon.

Lorsqu'un serveur X ne fonctionne pas avec une carte vidéo, les conséquences sont visibles: soit le serveur ne démarre pas du tout, soit l'affichage est incorrect.

Lorsqu'un client léger est prêt à démarrer en mode graphique, il appelle le script `startx`, qui lance le serveur X avec l'option **−query 'nom_de_serveur'**, où 'nom_de_serveur' correspond au serveur sur lequel un gestionnaire de connexion X **XDM**, **GDM** ou **KDM** est en cours d'exécution.

Si le serveur X est lancé par le script **startx**, il ne faut pas oublier que ce script est lui même lancé par le programme **init**. En cas d'erreur, **init** essaye systématiquement de relancer le script fautif (**startx** dans le cas précis), et ce 10 fois de suite. Après 10 essais infructueux, **init** abandonne, et on peut voir enfin les messages d'erreur du serveur X s'afficher à l'écran.

Il est plutôt agaçant d'attendre que le serveur X échoue 10 fois de suite afin de savoir pourquoi! Il existe un moyen d'éviter cette attente, en démarrant le client léger en runlevel (niveau d'exécution) 3. Dans cette configuration, le serveur X n'est PAS lancé automatiquement par **init**. Le client léger ouvre une session shell **bash** et attend qu'une commande soit tapée au clavier. A partir de là, on peut lancer manuellement le serveur X en appelant le script suivant:

```
sh  /tmp/start_ws
```

Le serveur X se lance et lorsqu'il échoue, le contrôle retourne immédiatement à la ligne de commande du shell. On peut alors vérifier pourquoi.

7.6. Gestionnaire de connexion X

Le gestionnaire de connexion X est un daemon lancé sur le serveur. Son rôle est d'attendre qu'un serveur X lui envoie une demande de connexion. Une fois le contact établi, il propose une fenêtre d'identification (login) sur l'écran, afin que l'utilisateur puisse se connecter sur le serveur.

Les trois gestionnaires de connexion les plus courants sont :

- XDM − C'est "l'ancêtre", disponible depuis l'origine de X Windows.
- GDM − ('Gnome Display Manager'). C'est le gestionnaire de connexion qui fait partie du package Gnome.
- KDM − ('KDE Display Manager'). C'est le gestionnaire de connexion qui fait partie du package KDE 'K Desktop system'.

La plupart des distributions Linux récentes contiennent ces trois gestionnaires de connexion X.

7.6.1. Le syndrome de l'écran gris et curseur en croix

Si cet écran gris s'affiche sur le client léger, cela signifie que le serveur X fonctionne bien, mais qu'il n'a pu entrer en contact avec un gestionnaire de connexion. Les causes probables de ce symptôme sont :

1. Le gestionnaire de connexion n'est pas lancé

   Sur les serveurs Redhat récents (7.0 et au delà), le gestionnaire de connexion est lancé à partir d'**init**. Dans le fichier `/etc/inittab`, on trouve une ligne qui ressemble à celle ci :

   ```
   x:5:respawn:/etc/X11/prefdm −nodaemon
   ```

   C'est le script **prefdm** qui détermine quel est le gestionnaire de connexion à lancer.

   Le gestionnaire de connexion par défaut dépend des packages installés sur le serveur. Si Gnome est installé, c'est GDM qui est le gestionnaire de connexion par défaut. Si Gnome n'est pas installé, le script **prefdm** vérifie si KDE est installé, et dans ce cas, c'est KDM qui sera le gestionnaire de connexion par défaut. Si KDE n'est pas non plus installé, c'est XDM qui sera le gestionnaire de connexion par défaut.

   En utilisant la commande **netstat**, il est possible de vérifier si un gestionnaire de connexion est lancé:

   ```
   netstat −ap | grep xdmcp
   ```

   Le résultat doit montrer qu'un processus est à l'écoute sur le port XDMCP (177).

   ```
   udp     0   0 *:xdmcp            *:*               1493/gdm
   ```

   Cet exemple montre que **gdm** est bien en cours d'exécution, que son PID est 1493, et qu'il est à l'écoute sur le pour XDMCP.

Si une ligne équivalente à celle ci apparaît, c'est qu'il y a un gestionnaire de connexion X lancé. Il faut donc vérifier que le client léger envoie bien une requête XDMCP à ce gestionnaire de connexion.

Dans le fichier `lts.conf`, il existe un paramètre (optionnel) qui permet d'indiquer l'adresse IP du serveur sur lequel le gestionnaire de connexion est lancé:

```
XDM_SERVER  =  192.168.0.254
```

Si ce paramètre est présent, sa valeur doit bien sûr correspondre à l'adresse IP du serveur concerné.

Si le paramètre 'XDM_SERVER' n'est pas défini, c'est la valeur du paramètre 'SERVER' qui est utilisée. Et si le paramètre 'SERVER' n'est pas non plus défini, la valeur par défaut utilisée est **192.168.0.254**.

Quelle que soit la façon de paramétrer l'adresse IP du serveur où le gestionnaire de connexion est lancé, il faut s'assurer que cette adresse est correcte.

2. Le gestionnaire de connexion est peut−être configuré pour refuser les connexions de machines distantes.

   Si les vérifications précédentes ont été faites et que cela ne fonctionne toujours pas, il se peut alors que le gestionnaire de connexion X soit configuré pour refuser les requêtes XDMCP provenant de machines distantes. Il faut donc vérifier les fichiers de configuration spécifiques à chaque gestionnaire de connexion.

   - **XDM**

     Sur les serveurs Redhat, XDM est configuré par défaut pour refuser les connexions distantes. Le script **ltspcfg** devrait normalement modifier ce paramétrage, mais si cela ne fonctionne pas, il faut mettre à jour le fichier `/etc/X11/xdm/xdm−config` manuellement. Rechercher si la ligne suivante existe dans le fichier :

     ```
     DisplayManager.requestPort:     0
     ```

     Cette ligne DOIT être mise en commentaire pour XDM accepte des requêtes de machines distantes sur le port 177. Pour ce faire, insérer un '#' en début de ligne.

     Un autre fichier de configuration important pour XDM est `/etc/X11/xdm/Xaccess`. Il DOIT contenir une ligne commençant par le caractère '*' (astérisque). Sur les serveurs Redhat, cette ligne est mise en commentaire par défaut. Le script **ltspcfg** doit normalement enlever le commentaire, mais si ça ne fonctionne pas, il faut le faire manuellement, ou ajouter la ligne si elle n'existe pas du tout dans le fichier. Une fois modifiée ou rajoutée, cette ligne doit ressembler à ce qui suit :

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

   - **KDM**

     Les versions récentes de KDM utilisent un fichier de configuration appelé **kdmrc**. Selon les distributions Linux, ce fichier peut être stocké dans différents endroits. Pour une Redhat 7.2, on le trouve dans `/etc/kde/kdm/kdmrc`. Pour les autres distributions, utiliser la commande **locate** pour trouver où est stocké ce fichier.

Le fichier **kdmrc** contient une section **[Xdmcp]** section. Pour que KDM accepte les requêtes distantes XDMCP, la valeur du paramètre **Enable** de cette section doit être **true** (vrai).

Les anciennes versions de KDM utilisent plutôt le fichier de configuration de XDM, `/etc/X11/xdm`.

- **GDM**

  GDM utilise des fichiers de configuration différents. Ils sont stockés dans le répertoire `/etc/X11/gdm`.

  Le principal fichier à contrôler est `gdm.conf` file. Dans la section **[xdmcp]**, le paramètre 'Enable' doit avoir la valeur '1' ou 'true', selon la version de GDM. Exemple:

  ```
  [xdmcp]
  Enable=true
  HonorIndirect=0
  MaxPending=4
  MaxPendingIndirect=4
  MaxSessions=16
  MaxWait=30
  MaxWaitIndirect=30
  Port=177
  ```

  La valeur de 'Enable' est bien 'true'. Dans les versions plus anciennes de GDM, on utilise '0' (faux) ou '1' (vrai) pour désactiver ou activer les connexions distantes. Les versions récentes utilisent 'false' et 'true'.

3. Si le problème persiste en dépit de toutes ces vérifications, il se peut enfin que le gestionnaie de connexion ne puisse pas assigner un nom d'hôte à l'adresse IP du client léger. Il faut alors contrôler que le nom du client léger est bien spécifié dans le fichier `/etc/hosts` file, ou qu'il est correctement géré dans les tables DNS (si DNS est utilisé sur le serveur).

Chapitre 8. Kernels

Il y a quelques décisions à prendre concernant le choix du noyau Linux pour un client léger. On peut soit decider d'utiliser un des noyaux standard disponibles en téléchargement, soit en compiler un soi−même. On peut aussi choisir de faire afficher, pendant le chargement du noyau, un écran graphique avec une barre de progression, ce qui est désormais possible avec le patch (modification) **Linux Progress Patch (LPP)**.

8.1. Noyaux standards fournis avec LTSP

Selon la version de LTSP, le package des noyaux contient 2 noyaux. Le premier a le patch 'Linux Progress Patch' déjà appliqué et configuré, le second ne l'a pas.

Les 2 noyaux sont déjà modifiés avec le patch 'NFS Swap patch'.

8.2. Compiler son propre noyau

Lorsqu'on décide de compiler un nouveau noyau pour les clients légers, il faut opter pour l'une des deux méthodes proposées ci−dessous.

La méthode par défaut utilise une solution basée sur un disque virtuel en mémoire, appelé "Initial Ram Disk", et plus connu sous l'acronyme **initrd**. L'image de ce disque virtuel initrd est un petit système de fichier qui est rajouté à la fin du noyau. Cette image initrd est chargée en mémoire, et lorsque le noyau démarre, il monte ce disque virtuel à la racine de son système de fichier (/). Il y a plusieurs avantages à utiliser cette technique. Tout d'abord, les drivers de cartes réseau peuvent être compilés sous forme de modules et seul le module nécessaire sera chargé et utilisé par le noyau au démarrage. Ceci permet donc d'utiliser le même noyau avec n'importe quelle carte réseau. L'autre avantage est que le client DHCP chargé par le client léger fonctionne dans l'espace utilisateur (user space) plutôt que dans l'espace noyau (kernel space). Cela permet de mieux contrôler les paramètres demandés et envoyés par le serveur, et permet enfin de réduire sensiblement la taille du noyau. L'autre méthode consiste à utiliser un noyau sans initrd. Dans ce cas, le noyau doit être compilé avec le driver spécifique à la carte réseau du client léger. Cela nécessite également d'utiliser les options "IP−Autoconfig" (Auto configuration IP) et "Root filesystem on NFS" (racine du système de fichier via NFS) pour compiler le noyau. Le fichier contenant le noyau sera plus petit (puisqu'il n'y a pas d'initrd rajouté), et le téléchargement depuis le serveur sera légèrement plus rapide. Mais une fois que le client léger a démarré et que l'initialisation du noyau est terminé, il n'y aura plus aucune différence entre ces deux méthodes quant au fonctionnement du client léger.

Le noyau standard LTSP contient une image initrd contenant tout ce qui est nécessaire à la détection de la carte réseau et à la gestion des requêtes DHCP dans l'espace utilisateur. L'objectif principal était de faire le plus petit initrd possible. Pour ce faire, nous avons choisi la librairie uClinux en remplacement de la librairie classique libc, et busybox pour remplacer tous les utilitaires standard utilisés pendant la phase de démarrage.

Si vous voulez compiler votre propre noyau, vous devrez télécharger le package `ltsp_initrd_kit`. Il contient l'arborescence du système de fichier racine, ainsi que les scripts pour construire l'image initrd.

8.2.1. Récupérer les sources du noyau Linux

Pour contruire un nouveau noyau, il est recommandé de commencer avec une version originale des sources, directement téléchargée depuis le site **ftp.kernel.org**. En effet, la plupart des distributions utilisent un noyau modifié par de nombreux patches (modifications), et dont le code source ne correspond par à celui des noyaux

officiels.

Téléchargez le source du noyau de votre choix et sauvegardez−le dans le répertoire `/usr/src` de votre serveur. Les noyaux officiels sont dans le repertoire `/pub/linux/kernel` du serveur **ftp.kernel.org**. Choisissez un noyau récent de la série 2.4, car le support de **devfs** est requis pour LTSP.

Si vous voulez également inclure le support du swap via NFS et le Linux Progress Patch (LPP), il faut se procurer les fichiers de patch (modifications) correspondants exactement à la version du noyau choisi. A l'heure où cet article est écrit, le noyau 2.4.9 est le dernier à supporter ces fonctionnalités.

Dans l'exemple qui suit, nous utiliserons le noyau 2.4.9. Le fichier contenant les sources est `ftp://ftp.kernel.org/pub/linux/kernel/v2.4/linux−2.4.9.tar.bz2`

Décompressez les sources du noyau dans le répertoire `/usr/src`. Attention, le répertoire `linux` sera alors créé. Si vous aviez déjà un tel répertoire contenant les sources d'un autre noyau, renommez−le avant de décompressez les sources du nouveau noyau.

Le package de sources que nous venons de télécharger ayant été compressé avec l'utilitaire **bzip2**, nous devons le décompresser avant de le passer à la commande **tar**. Utilisez la commande suivante pour décompresser :

```
bunzip2 <linux−2.4.9.tar.bz2 | tar xf −
```

Lorsque la décompression est terminée, le répertoire `/usr/src/linux` contient l'arborescence complète des sources du noyau Linux. A ce stade, il est recommandé de renommer ce répertoire plus significativement, en fonction de la version.

```
mv linux linux−2.4.9
```

Une fois renommé, rentrez dans ce répertoire :

```
cd linux−2.4.9
```

Avant de configurer le noyau, je modifie le fichier `Makefile`. Dans les premières lignes de ce fichier, on trouve une variable appelée **EXTRAVERSION**. Je donne la valeur 'ltsp−1' à cette variable. De cette manière, le véritable numéro de version de ce noyau sera '2.4.9−ltsp−1', ce qui facilitera son identification plus tard. Après modification, le début du fichier Makefile devrait ressembler à ce qui suit :

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 9
EXTRAVERSION = −ltsp−1

KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

8.2.2. Les patches Noyau

Après décompression, il se peut que vous deviez appliquer quelques patches (modifications) sur le code source, comme par exemple le NFS Swap Patch ou le Linux Progress Patch. Ces modifications doivent être faites AVANT la configuration du noyau.

8.2.2.1. NFS Swap patch

Le NFS Swap patch permet à un client léger d'utiliser un fichier de swap (pagination) se trouvant sur un serveur NFS distant. Bien qu'il soit recommandé de disposer de suffisament de mémoire RAM pour éviter la pagination, il peut être parfois difficile d'en installer en supplément, en particulier sur les vieux ordinateurs recyclés en clients légers. Utiliser un fichier de pagination via NFS permet de contourner ce problème.

Si le répertoire courant est `/usr/src/linux−2.4.9`, et que le fichier patch est dans `/usr/src`, vous pouvez utiliser la commande suivante pour tester le patch sans l'appliquer :

```
patch −p1 −−dry−run <../linux−2.4.9−nfs−swap.diff
```

Ceci va vérifier que le patch peut être appliqué sans problème. Si la commande ne provoque pas d'erreur, on peut alors appliquer la modification sur les sources, en enlevant l'option **−−dry−run** :

```
patch −p1 <../linux−2.4.9−nfs−swap.diff
```

8.2.2.2. Linux Progress Patch (LPP)

Le Linux Progress Patch (LPP) permet de faire afficher un logo graphique pendant que le noyau démarre. Les messages habituels du noyau sont redirigés vers un autre écran tty. Ce patch modifie le source des scripts de démarrage, qui feront évoluer une barre de progression au fur et à mesure de leur exécution.

Comme pour le NFS Swap patch, vous pouvez tester le patch avec la commande:

```
patch −p1 −−dry−run <../lpp−2.4.9
```

Si le test est concluant, appliquez le patch:

```
patch −p1 <../lpp−2.4.9
```

8.2.3. Configuration des options du noyau

Parmi les suivants, vous pouvez utiliser le programme de configuration noyau de votre choix :

- make xconfig

  Cette commande appelle la version X Windows de l'utilitaire de configuration du noyau.

- make menuconfig

  Cette commande appelle la version 'curses' (menu mode caractère) de l'utilitaire de configuration du noyau.

- make config

  Cette commande appelle une version de l'utilitaire de configuration du noyau qui affiche les paramètres ligne après ligne.

8.2.3.1. Configuration du noyau pour utilisation avec initrd

Configurer un noyau pour utilisation avec initrd nécessite d'activer les options suivantes :

- File systems −> /dev filesystem support

  Le support du système de fichiers `/dev` doit être activé. Cette option se trouve dans la section 'File systems'. N'activez PAS l'option 'Automatically mount at boot', ce montage étant réalisé par le script **/linuxrc**.

- Block devices −> RAM disk support

  Les clients légers utilisent un disque virtuel. Activez cette option, qui se trouve dans la section 'Block devices'.

- Block devices −> Initial RAM disk (initrd) support

  Pour initrd, cette option DOIT aussi être activée.

- Processor type and features −> Processor family

  Assurez vous que le noyau que vous configurez sera bien compilé pour le processeur (CPU) du client léger. Choisissez le processeur cible dans la section 'Processor type and features'. Vous devez aussi désactiver le support SMP (multi−processeur), à moins que votre client léger en possède plus d'un.

- File systems −> Network file systems −> NFS Client support

  Le client léger devra monter la racine de son système de fichier via NFS. En conséquence, le support (client) de NFS est requis.

Ceci termine la liste des options particulières requises par LTSP. Mais vous pouvez également désactiver de nombreuses autres options, afin de réduire la taille du noyau généré.

8.2.3.2. Configuration d'un noyau sans initrd

Il y a quelques différences de configuration entre un noyau utilisant initrd et un autre ne s'en servant pas.

- Block devices −> RAM disk support

  Les clients légers utilisent les disques virtuels en RAM. Activez cette option.

- Block devices −> Initial RAM disk (initrd) support

  Cette option doit être DESactivée.

- Networking options −> IP:kernel level autoconfiguration

  Cette option doit être activée. Dès le démarrage, le noyau configurera l'interface ethernet eth0 en fonction des paramètres passés sur la ligne de commande du noyau..

Il n'est pas nécessaire d'activer les options DHCP, BOOTP ou RARP du noyau, car c'est le code Etherboot qui effectue ce type de requêtes avant que le noyau soit chargé, et qui place les paramètres IP ainsi récupérés sur la ligne de commande du noyau, qui n'a donc plus besoin de lancer ces requêtes.

- Network device support −> Ethernet (10 or 100Mbit)

  Quand initrd n'est PAS utilisé, vous devez choisir et activer le driver de la carte réseau du client léger concerné. Ce driver DOIT être lié (linké) statiquement au noyau, car l'accès au réseau est nécessaire pour monter la racine du système de fichier. C'est la principale différence entre un noyau utilisant initrd et un autre sans.

- File systems −> /dev filesystem support

  A partir de LTSP version 2.09pre2, le support de **devfs** est requis, que le noyau utilise initrd ou non.

- File systems −> Automatically mount at boot

  Lorsque initrd n'est PAS utilisé, le système de fichier `/dev` doit être monté par le noyau pendant sa phase de démarrage. Activez cette option en répondant 'Y'.

- File systems −> Network file systems −> NFS Client support

  Puisque la racine du système de fichier est montée via NFS, le support du client NFS dans le noyau est requis. Activez cette option.

8.2.4. Compilation du noyau

Pour vous simplifier le travail, le package `ltsp_initrd_kit` contient un fichier de configuration `.config` initial. Vous pouvez copier ce fichier dans le répertoire `/usr/src/linux−2.4.9` avant de configurer votre noyau.

Lorsque le noyau aura été configuré en activant ou désactivant les options disponibles, vous pouvez alors compiler et lier le noyau. Pour ce faire, utilisez les commandes suivantes dans l'ordre indiqué :

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

Vous pouvez également lier ces opérations dans une même chaîne de commandes :

```
make dep && make clean && make bzImage && make modules && make modules_install
```

Le double symbole & indique au shell de n'exécuter la commande suivante que lorsque la précédente s'est correctement terminée, et ainsi de suite.

Lorsque la compilation (et l'édition de liens) est terminée, le nouveau noyau est disponible dans le répertoire `/usr/src/linux−2.4.9/arch/i386/boot/bzImage`.

8.2.5. Préparation nouveau noyau pour Etherboot

Pour qu'Etherboot puisse télécharger le nouveau noyau, ce dernier doit être préparé. Cette opération est connue sous le nom de 'Tagging'. Elle a pour effet d'ajouter un code exécutable au début du noyau, qui sera exécuté avant que le contrôle soit passé au noyau proprement dit. L'outil pour préparer un noyau pour Etherboot se nomme '**mknbi−linux**'.

Le package `tsp_initrd_kit` contient un script shell appelé **buildk** qui effectue toutes les commandes nécessaires à la préparation d'un noyau pour son téléchargement par le réseau.

Chapitre 9. Paramètres du fichier `lts.conf`

Lorsque nous avons conçu LTSP, nous savions que le principal défi consistait à gérer un très grand nombre de configurations matérielles pour les clients légers. Sans compter que les combinaisons de processeurs, de cartes réseau et de cartes vidéo disponibles aujourd'hui ne seraient peut être plus les mêmes quelques mois plus tard, au moment d'ajouter de nouveaux clients légers au réseau.

Nous avons donc cherché un moyen de spécifier la configuration de chaque client léger. Ces configurations sont définies dans le fichier `lts.conf` qui se trouve dans le répertoire `/opt/ltsp/i386/etc`.

Le format de `lts.conf` autorise la définition d'options 'par défaut' et d'autres spécifiques à des clients légers particuliers. Si tous vos clients légers sont identiques, vous n'aurez ainsi à configurer qu'un seul modèle dans la section '[Default]'.

9.1. Exemple de fichier `lts.conf`

Voici un exemple de fichier `lts.conf` :

```
[Default]
        SERVER             = 192.168.0.254
        X_MOUSE_PROTOCOL   = "PS/2"
        X_MOUSE_DEVICE     = "/dev/psaux"
        X_MOUSE_RESOLUTION = 400
        X_MOUSE_BUTTONS    = 3
        USE_XFS            = N
        SCREEN_01          = startx

[ws001]
        XSERVER            = auto
        X_MOUSE_PROTOCOL   = "Microsoft"
        X_MOUSE_DEVICE     = "/dev/ttyS1"
        X_MOUSE_RESOLUTION = 50
        X_MOUSE_BUTTONS    = 3
        X_MOUSE_BAUD       = 1200

[ws002]
        XSERVER            = XF86_Mach64

[ws003]
        SCREEN_01          = shell
```

9.2. Liste des paramètres `lts.conf`

9.2.1. Paramètres généraux

Commentaires
: Un commentaire commence par le caractère '#' et continue jusqu'à la fin de la ligne.

LTSP_BASEDIR
: Indique où se trouve (sur le serveur) la racine des systèmes de fichiers des clients légers. La valeur par défaut est `/opt/ltsp`

SERVER

Indique l'adresse du serveur XDM_SERVER, TELNET_HOST, XFS_SERVER et SYSLOG_HOST, si aucun de ceux−là n'est explicitement spécifié ailleurs. Si vous avez un serveur offrant simultanément tous ces services, indiquez son adresse ici, et vous pouvez omettre tous les autres. Par défaut (quand non spécifié), SERVER prend la valeur **192.168.0.254**.

SYSLOG_HOST

Si vous souhaitez envoyer les messages de journalisation syslog d'un client léger vers une machine qui n'est pas le serveur par défaut, vous pouvez indiquer son adresse ici. Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre 'SERVER' défini ci−dessus.

NFS_SERVER

Permet de spécifier l'adresse du serveur NFS utilisé pour monter le système de fichier `/home` d'un client léger (voir aussi: Exécution locale des applications). Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre 'SERVER'.

USE_NFS_SWAP

Si vous voulez utiliser le swap (pagination) via NFS, indiquez la valeur 'Y'. Valeur par défaut : **N**

SWAPFILE_SIZE

Taille du fichier de pagination (swap). Valeur par défaut : **64m**.

SWAP_SERVER

Le fichier de swap d'un client léger peut être créé sur n'importe quel serveur du réseau capable de le gérer via NFS. Vous pouvez indiquer ici l'adresse IP de ce serveur. Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre 'NFS_SERVER' défini ci−dessus.

NFS_SWAPDIR

Répertoire du serveur exporté via NFS et dans lequel le fichier swap d'un client léger sera créé. Valeur par défaut: `/var/opt/ltsp/swapfiles`. Assurez vous que le nom de ce répertoire exporté est également spécifié dans le fichier `/etc/exports`.

TELNET_HOST

Si un client léger est configuré pour ouvrir une session telnet (mode caractère), ce paramètre permet de définir sur quel serveur le client léger doit ouvrir cette session. Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre **SERVER**.

DNS_SERVER

Utilisé pour construire le fichier `resolv.conf`.

SEARCH_DOMAIN

Utilisé pour construire le fichier `resolv.conf`.

SCREEN_01 à SCREEN_12

Jusqu'à 12 scripts d'écran peuvent être demandés pour un même client léger. Ceci créera jusqu'à à 12 sessions simultanées, auxquelles on accède en utilisant les combinaisons de touches Ctrl−Alt−F1 à Ctrl−Alt−F12.

```
SCREEN_01   = startx
SCREEN_02   = shell
```

Les scripts d'écran disponibles sont :

- startx
- telnet
- rdesktop
- shell

Voir aussi dans le répertoire `/opt/ltsp/i386/etc/screen.d` pour d'éventuels scripts supplémentaires. Vous pouvez d'ailleurs écrire vos propres scripts et les placer dans ce répertoire.

MODULE_01 à MODULE_10

Ces paramètres permettent de demander le chargement de 1 à 10 modules noyaux optionnels sur un client léger. Pour chacun de ces paramètres, vous pouvez indiquer n'importe quelle option reconnue par la commande 'insmod', comme par exemple :

```
MODULE_01   = uart401.o
MODULE_02   = "sb.o io=0x220 irq=5 dma=1"
MODULE_03   = opl3.o
```

Si une valeur spécifiée indique un chemin d'accès absolu, la commande **insmod** sera utilisée pour charger le module correspondant. Dans le cas contraire, c'est la commande **modprobe** qui sera utilisée.

RAMDISK_SIZE

Lorsque le client léger démarre, un disque virtuel est créé dans sa mémoire, puis monté sur `/tmp`. Vous pouvez contrôler la taille de ce système de fichier avec ce paramètre. L'unité utilisée est le Kilo−octet (1024 octets). Pour créer un disque virtuel de 1 Mo, indiquez **RAMDISK_SIZE = 1024**

Si vous changez cette valeur ici, vous devrez également la changer dans le paramétrage du noyau. Le noyau devra alors être recompilé, ou bien, dans le cas où vous utilisez Etherboot ou Netboot, il faudra re−préparer le noyau (tagging) avec l'outil **mknbi−linux** pour spécifier la nouvelle taille du RAMDISK..

Valeur par défaut: 1024 (1 Mo)

RCFILE_01 à RCFILE_10

Vous pouvez spécifier ici jusqu'à 10 scripts RC qui seront exécutés par le script **rc.local** du client léger. Copiez les scripts souhaités dans le répertoire `/etc/rc.d`, puis indiquez leur nom ici.

SOUND

Si le package LTSP Sound est installé, indiquez **Y** ici. Le script **rc.sound** sera alors exécuté pour initialiser la carte son du client léger et pour lancer le daemon correspondant. Valeur par défaut : **N**.

9.2.2. Paramètres X−Window

XDM_SERVER

Si vous souhaitez que les clients légers s'adressent à un gestionnaire de connexion XDM se trouvant sur un autre serveur que celui par défaut, vous pouvez spécifier son adresse IP ici. Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre **SERVER**.

XSERVER

Ce paramètre définit le type de serveur X qui sera exécuté sur le client léger. Pour les cartes vidéo PCI et AGP, ce paramètre ne devrait pas être nécessaire, car le script rc.local.script doit être capable de détecter automatiquement la carte vidéo du client léger. Vous pouvez aussi indiquer la valeur **auto** pour demander la détection automatique de la carte (valeur par défaut).

Pour les cartes vidéo ISA, ou pour forcer le chargement d'un serveur X particulier, vous pouvez spécifier ici un nom de driver vidéo ou le nom d'un serveur X.

Si la valeur commence par **XF86_**, le serveur XFree86 3.3.6 sera utilisé. Dans le cas contraire, c'est le serveur Xorg qui sera utilisé. Valeur par défaut : **auto**.

X_MODE_0** à **X_MODE_2

Il est possible de définir jusqu'à 3 modes vidéo pour un client léger. La valeur de ces paramètres peut utiliser 2 syntaxes différentes : soit une résolution d'écran, soit une ligne 'modeline' complète.

```
X_MODE_0 = 800x600

ou

X_MODE_0 = 800x600 60.75 800 864 928 1088 600 616 621 657 −HSync −VSync
```

Si aucun paramètre X_MODE_x entries n'est spécifié, le serveur X utilisera des lignes 'modeline' standard, et les résolutions d'écran 1024x768, 800x600 et 640x480.

Si un ou plusieurs X_MODE_x sont définis, ils écrasent et remplacent les valeurs standard.

X_MOUSE_PROTOCOL

N'importe quel mot−clé compatible avec la syntaxe de XFree86 pour les protocoles souris peut être utilisé ici, comme "Microsoft" ou "PS/2". Valeur par défaut : **"PS/2"**.

X_MOUSE_DEVICE

Indique le nom du périphérique (device) sur lequel la souris est connectée. Pour une souris série, c'est en général un port série comme `/dev/ttyS0` ou `/dev/ttyS1`. S'il s'agit d'une souris PS/2, il s'agit alors de `/dev/psaux`. Valeur par défaut : `/dev/psaux`.

X_MOUSE_RESOLUTION

On peut indiquer ici une valeur pour le paramètre 'Resolution' du fichier `XF86Config`. Pour une souris série, la valeur habituelle est **50** et **400** pour une souris PS/2. Valeur par défaut : **400**.

X_BUTTONS

Indique le nombre de bouton de la souris, en général **2** ou **3**. Valeur par défaut : **3**.

X_MOUSE_EMULATE3BTN

Indique au serveur X d'émuler une souris 3 boutons avec une souris à 2 boutons, en acceptant le clic simultané sur les boutons gauche et droit. Valeur par défaut : **N**.

X_MOUSE_BAUD

Définition de la vitesse de communication pour une souris série. Exprimée en bauds. Valeur par défaut : **1200**.

X_COLOR_DEPTH

Bits utilisés pour le nombre de couleurs. Valeurs possibles : **8**, **15**, **16**, **24** et **32**. 8 bits permettent la gestion de 256 couleurs, 16 bits 65536 couleurs, 24 bits 16 millions de couleurs et 32 bits donneront 4.2 milliards de couleurs! Certains serveurs X n'acceptent pas toutes ces valeurs. Valeur par défaut : **16** (65536 couleurs)

USE_XFS

Pour que le client léger récupère ses polices de caractères il y a deux possibilités : soit les demander à un serveur de polices XFS (X Font Server) du réseau, soit les lire directement sur son système de fichier monté par NFS. Un serveur de polices permet de gérer simplement toutes les polices sur une seule machine du réseau, mais quelques problèmes peuvent surgir lorsque le nombre de clients légers dépasse les 40. Les 2 valeurs autorisées pour ce paramètres sont **Y** et **N**. Valeur par défaut : **N**. Si vous voulez vraiment utiliser un serveur de polices, indiquez 'Y' ici, et donnez l'adresse IP de ce serveur dans le paramètre **XFS_SERVER**.

XFS_SERVER

Lorsqu'un serveur de polices XFS est utilisé, ce paramètre permet d'en spécifier l'adresse IP. Si ce paramètre n'est PAS spécifié, il prend la valeur du paramètre **SERVER**.

X_HORZSYNC

Indique la valeur du paramètre XFree86/Xorg **HorizSync**. Valeur par défaut : **"31−62"**.

X_VERTREFRESH

Indique la valeur du paramètre XFree86/Xorg **VertRefresh** Valeur par défaut : **"55−90"**.

XF86CONFIG_FILE

Si vous souhaitez utiliser un fichier de configuration X 'XF86Config' complet, copiez le dans le répertoire `/opt/ltsp/i386/etc`. Quel que soit son nom, il doit être indiqué ici. Par exemple :

```
XF86CONFIG_FILE = XF86Config.ws004
```

9.2.3. Paramètres pour écrans tactiles

USE_TOUCH

Si vous connectez un écran tactile sur un client léger, il faut alors l'activer en indiquant **Y** ici. Dans ce cas, les autres options spécifiques aux écrans tactiles seront prises en compte. Valeur par défaut : **N**.

X_TOUCH_DEVICE

Un écran tactile fonctionne comme une souris. Il est généralement connecté au client léger par un port série, dont vous préciserez le numéro ici, par exemple, `/dev/ttyS0`. Ce paramètre n'a pas de valeur par défaut.

X_TOUCH_MINX

Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **433**.

X_TOUCH_MAXX

Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **3588**.

X_TOUCH_MINY

Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **569**.

X_TOUCH_MAXY

Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **3526**.

X_TOUCH_UNDELAY

Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **10**.

X_TOUCH_RPTDELAY
Etalonnage des écrans tactiles EloTouch. Valeur par défaut : **10**.

9.2.4. Paramètres pour support des applications locales

LOCAL_APPS
Si vous souhaitez exécuter vos applications sur les clients légers plutôt que sur le serveur, indiquez ici la valeur **Y**. D'autres opérations doivent être effectuées avant que le support d'applications locales soit opérationnel. Reportez vous à la section "Applications locales' de ce manuel LTSP pour plus d'informations. Valeur par défaut : **N**.

NIS_DOMAIN
Si vous avez activé le support des applications locales avec le paramètre LOCAL_APPS, un serveur NIS doit être présent et actif sur le réseau. Le paramètre NIS_DOMAIN permet de spécifier le nom du domaine NIS géré par ce serveur. Notez qu'un domaine NIS n'a RIEN de commun avec un domaine DNS − Internet. Valeur par défaut : **ltsp**.

NIS_SERVER
Si vous voulez éviter que vos clients légers envoient des broadcasts afin de rechercher un serveur NIS, indiquez ici son adresse IP.

9.2.5. Paramètres pour gestion du clavier

Tous les fichiers nécessaires à la gestion des claviers des clients légers sont désormais stockés dans l'arborescence `/opt/ltsp/i386`. Configurer le clavier d'un client léger LTSP consiste en fait à configurer ce clavier pour XFree86. Plusieurs paramètres sont disponibles.

Les valeurs autorisées pour les paramètres ci−dessous proviennent de la documentation XFree86. Toute valeur acceptée par XFree86 peut être spécifiée pour ces paramètres.

Nous souhaiterions pouvoir ajouter ici plus d'information concernant la gestion des différents claviers nationaux. Si vous avez l'habitude de configurer de tels claviers, merci de faire part de votre expérience au groupe de développement de LTSP.

XkbTypes
La valeur par défaut est le mot '**default**'.

XkbCompat
La valeur par défaut est le mot '**default**'.

XkbSymbols
La valeur par défaut est le mot '**default**'.

XkbModel
La valeur par défaut est le mot '**pc101**'. (ndt : pour un clavier français il faut **pc105**)

XkbLayout
La valeur par défaut est le mot '**us**'. (ndt : pour clavier français il faut **fr**)

9.2.6. Paramètres de configuration des imprimantes

De une à trois imprimantes peuvent être connectées à un client léger, sur un port série ou parallèle. Les paramètres suivants du fichier `lts.conf` permettent de les configurer.

PRINTER_0_DEVICE
: Nom du périphérique (device) pour la première imprimante. Des noms comme `/dev/lp0`, `/dev/ttyS0` ou `/dev/ttyS1` sont autorisés.

PRINTER_0_TYPE
: Type de la première imprimante. Valeurs autorisées : '**P**' pour Parallèle, et '**S**' pour Série.

PRINTER_0_PORT
: Numéro du port TCP/IP utilisé pour la première imprimante. Valeur par défaut : '**9100**'.

PRINTER_0_SPEED
: Si la première imprimante est connectée en série, ce paramètre précise la vitesse de transmission, en bauds. Valeur par défaut : '**9600**'.

PRINTER_0_FLOWCTRL
: Si la première imprimante est connectée en série, ce paramètre précise le type de gestion de flux (flow control). Valeurs autorisées : '**S**' pour gestion 'Software' (XON/XOFF) ou '**H**' pour gestion 'Hardware' (CTS/RTS). Valeur par défaut : '**S**'.

PRINTER_0_PARITY
: Si la première imprimante est connectée en série, ce paramètre précise la parité utilisée. Valeurs autorisées : '**E**' (Parité Paire), '**O**' (Impaire) ou '**N**' (Pas de parité). Valeur par défaut : '**N**'.

PRINTER_0_DATABITS
: Si la première imprimante est connectée en série, ce paramètre précise le nombre de bits de données. Valeurs autorisées : '**5**', '**6**', '**7**' et '**8**'. Valeur par défaut : '**8**'.

PRINTER_1_DEVICE
: Périphérique de la deuxième imprimante

PRINTER_1_TYPE
: Type de la deuxième imprimante

PRINTER_1_PORT
: Numéro de port TCP/IP de la deuxième imprimante

PRINTER_1_SPEED
: Vitesse de la deuxième imprimante (si série)

PRINTER_1_FLOWCTRL
: Gestion de flux de la deuxième imprimante (si série)

PRINTER_1_PARITY
: Parité de la deuxième imprimante (si série)

PRINTER_1_DATABITS

Bits de données de la deuxième imprimante (si série)

PRINTER_2_DEVICE
Périphérique de la troisième imprimante

PRINTER_2_TYPE
Type de la troisième imprimante

PRINTER_2_PORT
Numéro de port TCP/IP de la troisième imprimante

PRINTER_2_SPEED
Vitesse de la troisième imprimante (si série)

PRINTER_2_FLOWCTRL
Gestion de flux de la troisième imprimante (si série)

PRINTER_2_PARITY
Parité de la troisième imprimante (si série)

PRINTER_2_DATABITS
Bits de donénes de la troisième imprimante (si série)

Chapitre 10. Applications locales

Dans un environnement LTSP, il est possible de faire exécuter les applications soit sur le serveur, soit sur le client léger.

La méthode la plus simple, et de loin, est de configurer LTSP pour que toutes les applications s'exécutent sur le serveur. C'est d'ailleurs la configuration par défaut. Dans cette configuration, une application lancée par un client léger est exécutée sur le serveur, en utilisant le processeur et la mémoire de ce dernier. L'affichage de l'application est renvoyée sur l'écran du client léger, tandis que caractères et actions du clavier et de la souris du client léger sont renvoyés à l'application.

C'est une fonctionnalité fondamentale du système X Window. Le client léger fonctionne simplement comme un terminal X Window standard.

Mais lorsqu'on souhaite exécuter une application sur le client léger (on parle alors d'application 'locale'), ce dernier doit d'abord obtenir des informations sur l'utilisateur, comme :

- son numéro d'identifiant Linux (User id).
- Le groupe primaire (primary group) auquel l'utilisateur appartient.
- Le répertoire personnel (home directory) de l'utilisateur.

LTSP se base sur le service NIS (Network Information Service, anciennement nommé '***Pages Jaunes***') pour récupérer ces informations utilisateur et les communiquer au client léger.

10.1. Avantages de l'exécution locale d'applications

Il y a quelques avantages à exécuter localement les applications sur les clients légers.

- Cela réduit la charge du serveur. Sur des réseaux importants, lorsque des applications gourmandes en mémoire sont utilisées, comme Netscape, l'exécution locale sur un client léger peut améliorer les performances, du moins si le client léger est capable de gérer la surcharge correspondante.
- Une application qui plante sur un client léger n'affectera pas les autres utilisateurs.
- La gestion du son est bien plus facile à configurer lorsque l'application qui génère le son s'exécute sur le client léger.

10.2. Inconvénients de l'exécution locale d'applications

Le support des applications locales nécessite en revanche des pré−requis majeurs.

- La charge sera plus importante sur le client léger. Il faudra plus de mémoire RAM et un processeur plus puissant. On peut considérer qu'il faut au minimum 64 Mo de mémoire pour envisager l'exécution d'une application sur un client léger.
- Utilisation du service NIS. Pour exécuter localement une application, l'utilisateur doit d'abord s'identifier vis à vis du client léger. En d'autres termes, le système d'exploitation du client léger (Linux) a besoin de savoir qui se connecte, et pour cela, une forme d'authentification par mot de passe est nécessaire. NIS a été choisi pour permettre cette authentification à travers le réseau.
- Des répertoires supplémentaires doivent être exportés via NFS depuis le serveur vers les clients légers.

- Les applications seront plus lentes à démarrer, car elles seront téléchargées sur le client léger depuis le serveur via NFS, causant par la même occasion un trafic réseau plus important. D'autre part, puisque chaque instance d'une même application s'exécutera sur son propre processeur (client léger), vous ne bénéficierez pas de la possibilité offerte par Unix−Linux de partager en mémoire le même code entre plusieurs instances d'une application, ce qui permet d'accélerer son démarrage dès qu'elle est invoquée une deuxième fois.

10.3. Configuration du serveur LTSP pour l'exécution locale des applications

10.3.1. Paramètres de `lts.conf`

Quelques paramètres de `lts.conf` doivent être (re)configurés pour le support des applications locales :

LOCAL_APPS
: Ce paramètre doit être initialisé à **Y**. Durant la phase de démarrage d'un client léger, les opérations suivantes vont se déclencher :

1. le répertoire `/home` du serveur sera exporté via NFS et monté sur l'arborescence du client léger.
2. Le fichier `/var/yp/nicknames` sera créé sur le client léger.
3. Le service **portmapper** sera lancé sur le client léger.
4. Le gestionnaire de daemon **xinetd** sera lancé sur le client léger.
5. Le fichier `/etc/yp.conf` sera créé sur le client léger.
6. La commande **domainname** sera lancée, avec pour argument la valeur du paramètre **NIS_DOMAIN** spécifiée dans `lts.conf`.
7. Le programme **ypbind** sera exécuté sur le client léger.

NIS_DOMAIN
: Avec NIS, tous les noeuds d'un réseau souhaitant faire partie d'un même domaine NIS doivent s'enregistrer sur ce domaine (qui n'a rien à voir avec un domaine DNS). Utilisez ce paramètre NIS_DOMAN pour indiquer le nom du domaine dans lequel les client légers seront enregistrés.

NIS_SERVER
: Pour accéder à un serveur de domaine NIS, on peut soit envoyer un broadcast sur le réseau et attendre une réponse, soit envoyer une requête à un serveur précis. Si vous avez choisi un serveur NIS particulier, vous pouvez préciser ici son adresse IP.

10.3.2. Network Information Service − NIS

NIS est un service de type Client−Serveur. Sur le serveur, un daemon (programme de service) attend les requêtes de ses clients (les clients légers dans notre cas). Ce daemon fonctionnant sur le serveur s'appelle **ypserv**.

Sur le client léger un processus appelé **ypbind** est lancé. Lorsque le client léger souhaite contrôler une information relative à un utilisateur, comme vérifier son mot de passe ou trouver son répertoire personnel, il va utiliser **ypbind** pour établir une connexion avec le daemon **ypserv** du serveur.

Si votre réseau utilise déjà NIS, il n'est pas nécessaire de lancer **ypserv** sur le serveur LTSP. Il suffit seulement de donner les valeurs correctes aux paramètres NIS_DOMAINNAME et NIS_SERVER dans le fichier `lts.conf`.

Mais si vous n'utilisez PAS encore NIS, vous devrez alors configurer le serveur LTSP pour qu'il lance le daemon **ypserv**.

Pour obtenir plus d'informations sur comment installer et configurer un serveur NIS, reportez vous à la documentation 'HOWTO' intitulée ***The Linux NIS(YP)/NYS/NIS+ HOWTO*** que vous trouverez sur le serveur web www.tldp.org (The Linux Documentation Project). D'autres sources d'informations vous sont données à la fin de ce document.

10.4. Configuration des applications

Pour qu'une application puisse être exécutée localement sur un client léger, vous devrez installer tous les composants qu'elle utilise dans un espace accessible par le client léger.

Dans les versions précédents de LTSP (2.08 et inférieures), un grand nombre de répertoires du serveur étaient exportés via NFS et montés sur les clients légers. Less répertoires du serveur comme `/bin`, `/usr/bin`, `/lib` et `/usr` étaient accessibles depuis les clients légers.

L'inconvénient de cette méthode est qu'elle ne fonctionne correctement que lorsque le serveur et le(s) client(s) légers(s) ont la même architecture. Et même dans ce cas, si le serveur est par exemple un Pentium II (i686), et le client léger un Pentium classique (i586), quelques problèmes pourraient apparaître, car le serveur disposera sans doute des librairies pour i686, mais pas forcément pour i386, i486 ou 1586.

Le meilleur moyen de gérer ce problème est de disposer d'une arborescence fichier complète et spécifique au(x) client(s) léger(s), complètement distincte de l'arborescence du serveur, et contenant tous les programmes exécutables et les librairies utilisés localement par ce(s) client(s) léger(s).

Configurer une application pour son exécution locale sur un client léger consiste donc à installer tous les composants nécessaires dans cette arborescence. Par exemple, vous trouverez sur le site LTSP le package 'local netscape'. Lorsqu'il est installé sur le serveur, ce package copie tous ses composants dans `/opt/ltsp/i386/usr/local/netscape`. Des éléments comme les classes java, les fichiers d'aide, les scripts et les programmes exécutables (binaires) sont stockés là pour utilisation par les clients légers.

Netscape n'a pas besoin de librairies système supplémentaires, et il n'y a donc rien à rajouter dans `/opt/ltsp/i386/lib`. Mais de nombreuses applications ont besoin d'autres librairies.

Mais comment identifier les librairies nécessaires à une application ? C'est ici que l'utilitaire **ldd** s'avère très utile.

A titre d'exemple, imaginons que vous souhaitiez exécuter localement l'application **gaim**. **gaim** est un client de messagerie en ligne d'AOL, qui vous permet de communiquer avec d'autres personnes connectées aux forums AOL.

La première chose à faire est de trouver le programme exécutable **gaim**. Sur un serveur Redhat 7.2, ce programme est dans le répertoire `/usr/bin`.

Une fois **gaim** trouvé, vous pouvez alors lancer l'utilitaire **ldd**:

```
[jam@server /]$ ldd /usr/bin/gaim
        libaudiofile.so.0    => /usr/lib/libaudiofile.so.0 (0x40033000)
        libm.so.6            => /lib/i686/libm.so.6 (0x40051000)
```

```
        libnsl.so.1          => /lib/libnsl.so.1 (0x40074000)
        libgnomeui.so.32     => /usr/lib/libgnomeui.so.32 (0x4008a000)
        libart_lgpl.so.2     => /usr/lib/libart_lgpl.so.2 (0x4015d000)
        libgdk_imlib.so.1    => /usr/lib/libgdk_imlib.so.1 (0x4016c000)
        libSM.so.6           => /usr/X11R6/lib/libSM.so.6 (0x40191000)
        libICE.so.6          => /usr/X11R6/lib/libICE.so.6 (0x4019a000)
        libgtk-1.2.so.0      => /usr/lib/libgtk-1.2.so.0 (0x401b1000)
        libdl.so.2           => /lib/libdl.so.2 (0x402df000)
        libgdk-1.2.so.0      => /usr/lib/libgdk-1.2.so.0 (0x402e3000)
        libgmodule-1.2.so.0  => /usr/lib/libgmodule-1.2.so.0 (0x40319000)
        libXi.so.6           => /usr/X11R6/lib/libXi.so.6 (0x4031d000)
        libXext.so.6         => /usr/X11R6/lib/libXext.so.6 (0x40325000)
        libX11.so.6          => /usr/X11R6/lib/libX11.so.6 (0x40333000)
        libgnome.so.32       => /usr/lib/libgnome.so.32 (0x40411000)
        libgnomesupport.so.0 => /usr/lib/libgnomesupport.so.0 (0x40429000)
        libesd.so.0          => /usr/lib/libesd.so.0 (0x4042e000)
        libdb.so.2           => /usr/lib/libdb.so.2 (0x40436000)
        libglib-1.2.so.0     => /usr/lib/libglib-1.2.so.0 (0x40444000)
        libcrypt.so.1        => /lib/libcrypt.so.1 (0x40468000)
        libc.so.6            => /lib/i686/libc.so.6 (0x40495000)
        libz.so.1            => /usr/lib/libz.so.1 (0x405d1000)
        /lib/ld-linux.so.2   => /lib/ld-linux.so.2 (0x40000000)
```

La liste générée par ldd contient le nom de toutes les librairies avec lesquelles **gaim** est dynamiquement lié.

La plupart des applications qui utilisent des librairies partagées se servent du chargeur dynamique **ld−linux** pour trouver et charger ces librairies. En revanche, d'autres applications les chargent 'manuellement' par l'intermédiaire de l'appel système **dlopen()**. Pour ces applications, **ldd** n'affichera rien. Dans ce cas, vous devrez utiliser l'utilitaire **strace** pour tracer l'exécution de l'application et rechercher ses appels à la fonction système **dlopen()**. **strace** vous donnera les arguments passés à chaque appel de cette fonction. L'un d'entre eux correspond au nom d'une librairie demandée par l'application.

Une fois que vous aurez reconstitué la liste des librairies nécessaires à une application, vous devrez copier ces librairies dans les répertoires correspondants de l'arborescence `/opt/ltsp/i386`.

10.5. Lancement des applications locales

Dans un environnement X Window, la machine sur laquelle un programme s'exécute est déterminée en fonction de celle où le gestionnaire graphique a été lancé. Si le gestionnaire graphique est lancé sur le serveur avec son affichage redirigé vers un client léger, alors toute application lancée sera aussi exécutée sur le même serveur et son affichage redirigé vers le même client léger.

Pour exécuter un programme sur le client léger alors que le gestionnaire graphique fonctionne sur le serveur, il faut donc utiliser une astuce. Dans le cas présent, il faut que le serveur demande au client léger de lancer le programme à sa place, et pour cela on utilise la commande **rsh**.

L'exemple qui suit montre comment utiliser rsh pour lancer **gaim** sur le client léger:

```
HOST=`echo $DISPLAY | awk -F: '{ print $1 }'`
rsh ${HOST} /usr/bin/gaim -display ${DISPLAY}
```

Ces commandes peuvent être saisies dans une fenêtre **xterm**, ou stockées dans un fichier script lancé à partir d'un icône placé sur le bureau.

Le lancement local de Netscape est réalisé de façon similaire, mais une variable d'environnement supplémentaire est nécessaire.

```
HOST=`echo $DISPLAY | awk −F: '{ print $1 }'`
rsh ${HOST} MOZILLA_HOME=/usr/local/netscape \
/usr/local/netscape/netscape −display ${DISPLAY}
```

Chapitre 11. Exemples de configuration

N'importe quel élément d'un client léger peut être configuré à l'aide des paramètres correspondants du fichier `lts.conf`, qui est normalement situé dans le répertoire `/opt/ltsp/i386/etc`.

11.1. Souris Série

Exemple de `lts.conf` pour la définition d'une souris série à 2 boutons :

```
X_MOUSE_PROTOCOL    = "Microsoft"
X_MOUSE_DEVICE      = "/dev/ttyS0"
X_MOUSE_RESOLUTION  = 400
X_MOUSE_BUTTONS     = 2
X_MOUSE_EMULATE3BTN = Y
```

11.2. Souris PS/2 à molette

Exemple de `lts.conf` pour la définition d'une souris de type Intellimouse :

```
X_MOUSE_PROTOCOL    = "IMPS/2"
X_MOUSE_DEVICE      = "/dev/psaux"
X_MOUSE_RESOLUTION  = 400
X_MOUSE_BUTTONS     = 5
X_ZAxisMapping      = "4 5"
```

11.3. Imprimante USB sur un client ThinkNic

Les clients légers de type ThinkNIC possèdent un port USB, sur lequel on peut brancher une imprimante. Voici un exemple de de `lts.conf` pour une telle configuration :

```
MODULE_01           = usb-ohci
MODULE_02           = printer
PRINTER_0_DEVICE    = /dev/usb/lp0
PRINTER_0_TYPE      = S
```

11.4. Forcer le chargement de XFree86 3.3.6 sur un client léger

Par défaut, les clients légers sont configurés pour XFree86 4.1.0 ou Xorg. Si vous souhaitez utiliser une version antérieure (X3.3.6 Xserver) sur un client léger particulier, assurez-vous d'abord que le package correspondant soit installé sur le serveur. Vous pourrez alors modifier le fichier `lts.conf`. Voici un exemple demandant d'utiliser le serveur XFree86 3.3.6 **SVGA** :

```
XSERVER             = XF86_SVGA
```

Chapitre 12. Autres sources d'information

12.1. Documentation en ligne

1. Page d'accueil du site LTSP

   www.LTSP.org

2. Documentation sur les stations sans disque 'Diskless−Nodes HOW−TO document for Linux'

   www.linuxdoc.org/HOWTO/Diskless−HOWTO.html

3. Page d'accueil du site Etherboot

   etherboot.sourceforge.net

4. Le site Rom−O−Matic

   www.Rom−O−Matic.net

5. Gestion de la souris sous XFree86

   www.xfree86.org/current/mouse.html

6. XFree86−Video−Timings−HOWTO

   www.linuxdoc.org/HOWTO/XFree86−Video−Timings−HOWTO.html

7. The Linux NIS(YP)/NYS/NIS+ HOWTO

   www.linuxdoc.org/HOWTO/NIS−HOWTO.html

12.2. Publications

1. Managing NFS and NIS
   Hal Stern
   O'Reilly & Associates, Inc.
   1991
   ISBN 0−937175−75−7

2. TCP/IP Illustrated, Volume 1
   W. Richard Stevens
   Addison−Wesley
   1994
   ISBN 0−201−63346−9

3.

X Window System Administrator's Guide
Linda Mui and Eric Pearce
O'Reilly & Associates, Inc.
1993
ISBN 0−937175−83−8
(Volume 8 of the The Definitive Guides to the X Window System)