LTSP − Linux Terminal Server Project − v4.1

James McQuillan

<jam@LTSP.org>

Martin Newiger

<mn@mn-ix.de>

Martin Herweg

<m.herweg@gmx.de>

Wolfgang Schweer

<schweer@cityweb.de>

Versionsgeschichte

Version 4.1.3−en	2004−06−20	Geändert durch: jam
Version 4.1.3−0−de	2005−11−05	Geändert durch: mn
Version 4.1.3−1−de	2005−11−06	Geändert durch: ws
Version 4.1.3−2−de	2005−11−22	Geändert durch: mn

GNU/Linux bietet eine gute Grundlage für den Einsatz von plattenlosen "Thin Clients". In erster Linie soll dieses Dokument zeigen, wie plattenlose "Thin Clients" mit LTSP eingerichtet und betrieben werden können. Allerdings werden auch viele Aspekte zum Thema plattenlose Arbeitsplatzrechner im Allgemeinen behandelt.

Inhaltsverzeichnis

Inhaltsverzeichnis

Einführung

Das LTS−Projekt ermöglicht es, auf einfache Weise preisgünstige Arbeitsplatzrechner als graphische oder textbasierte Terminals eines GNU/Linux−Servers zu benutzen.

In einer klassischen Büroumgebung finden sich auf jedem Schreibtisch relativ gut ausgestattete PCs, jeder mit eigener Festplatte von mehreren Gigabytes. Die Benutzer speichern ihre Daten lokal, Backups werden selten (wenn überhaupt) durchgeführt.

Macht es wirklich Sinn, auf jedem Schreibtisch solch einen Rechner stehen zu haben?

Unsere Antwort lautet: Nein.

Glücklicherweise gibt es eine Alternative. Wenn man LTSP benutzt, dann reichen PCs am unteren Ende der Preisskala. Die Festplatte, das Disketten− und CDROM−Laufwerk können entfernt werden. Lediglich eine Netzwerkkarte, von der der Rechner gebootet werden kann, muss eingebaut werden. Viele Netzwerkkarten besitzen einen Bootrom−Sockel, der geradezu darauf wartet, dass ein Bootrom eingesetzt wird.

Während der Bootphase erhält der plattenlose Client seine IP−Adresse, weitere Informationen und den Kernel vom Server. Danach wird das Root−Verzeichnis vom Server per NFS gemountet.

Die Workstation kann auf drei Weisen konfiguriert werden:

- **Graphische Oberfläche − X Window System**

  Unter dem X Window System kann die Workstation alle Anwendungsprogramme auf dem Server oder auf anderen Servern im Netzwerk benutzen.

- **Textbasierte Telnet−Sitzungen**

  Die Workstation kann bis zu neun Telnet−Sitzungen zum Server aufbauen. Jede Sitzung läuft auf einer virtuellen Konsole. Mit ALT−F1 bis ALT−F9 kann zwischen den einzelnen Konsolen hin− und her geschaltet werden.

- **Shell−Eingabeaufforderung**

  Die Workstation kann so konfiguriert werden, dass man direkt in eine bash−Shell auf der Konsole eingeloggt wird. Dies ist beim Debuggen von X−Window− oder NFS−Problemen sehr nützlich.

Das Tolle an der Sache ist, dass eine Menge von Workstations von einem einzigen GNU/Linux−Server bedient werden kann. Wie viele Workstations können es denn sein? Nun, das hängt von der Leistungsfähigkeit des Servers und den Anwendungen ab, die benutzt werden sollen.

Es ist durchaus nicht ungewöhnlich, 50 Workstations mit Mozilla und OpenOffice über einen Dual P4−2.4 (Xeon) mit 4GB RAM zu betreiben. Wir wissen, dass es geht. In der Tat ist der load−average−Wert selten über 1.0!

1. Disclaimer

Weder der Autor noch die Übersetzer, Distributoren oder jeder andere, der etwas in irgendeiner Form zu diesem Dokument beiträgt, sind in irgendeiner Weise verantwortlich zu machen für physikalische, finanzielle, moralische oder irgendwelche anderen Schäden, welche infolge der Vorschläge in diesem Text eintreten.

Kapitel 1. Die Theorie

Das Booten einer plattenlosen Workstation umfasst mehrere Schritte. Versteht man diese, ist es viel einfacher, auftretende Probleme zu lösen.

Es werden vier grundlegende Dienste benötigt, um eine LTSP−Workstation zu booten. Diese sind:

- DHCP
- TFTP
- NFS
- XDMCP

LTSP ist extrem flexibel. Jeder der oben genannten Dienste kann von einem einzigen Server oder von mehreren verschiendenen Servern zur Verfügung gestellt werden. In unserem Beispiel werden wir das einfache Setup beschreiben, welches aus einem einzigen Server besteht, der die Dienste bereitstellt.

1.1. Der Konfigurationsablauf auf der Workstation

1. Laden des Linux−Kernels in den Arbeitsspeicher der Workstation: Dies kann auf eine der folgenden Weisen passieren:

    a. Bootrom (Etherboot, PXE, MBA, Netboot)
    b. Floppy
    c. Festplatte
    d. CD−ROM
    e. USB−Speichergerät

   Jede der oben genannten Bootmethoden wird später in diesem Kapitel erklärt.

2. Sobald der Kernel in den Speicher geladen ist, wird er ausgeführt.

3. Der Kernel initialisiert das gesamte System und alle Peripheriegeräte, die von ihm erkannt werden.

4. Jetzt wird es richtig spannend. Während des Kernel−Ladeprozesses wird auch ein RAM−Disk−Image in den Speicher geladen; angehängt ist nämlich ein Image eines Dateisystems. Ein Kommandozeilen−Parameter in der Form **root=/dev/ram0** teilt dem Kernel mit, dass er dieses Image als Root−Dateisystem mounten soll.

5. Normalerweise führt der Kernel nach dem Booten das **init**−Programm aus. In diesem Fall jedoch weisen wir den Kernel an, stattdessen ein Shell−Skript zu laden. Dies geschieht durch den Parameter **init=/linuxrc/** in der Kommandozeile des Kernels.

6. Das **/linuxrc/**−Skript sucht zu Beginn durch Scannen des PCI−Busses nach einer Netzwerkkarte. Für jedes gefundene PCI−Device erfolgt ein Vergleich mit der Datei bekannter Netzwerkkarten (/etc/niclist). Falls dort ein Eintrag gefunden wird, wird das Netzwerkkarten−Treibermodul zurückgegeben und dieses wird abschließend auch geladen. Für ISA−Karten muss der Name des Moduls dem Kernel als Kommandozeilen−Parameter übergeben werden. Zusätzlich sind noch eventuell erforderliche Werte für IO−Adresse und IRQ zu übergeben.

7. Nun wird ein kleiner DHCP−Client namens **dhclient** aufgerufen, um eine erneute Anfrage an den

DHCP−Server zu senden. Wir führen diese Anfrage im User−Space durch, weil wir noch mehr Informationen benötigen als die, die das Bootrom nach der ersten DHCP−Anfrage erhalten hat.

8. Erhält das Programm **dhclient** eine Antwort vom Server, ruft es die Datei **/etc/dhclient−script/** auf, das die erhaltenen Informationen nutzt und somit das Netzwerkinterface eth0 konfiguriert.

9. Bis zu diesem Zeitpunkt lag das root−Dateisystem auf einer RAM−Disk vor. Jetzt wird das /linuxrc−Skript ein neues root−Dateisystem via NFS mounten. Typischerweise wird das Verzeichnis **/opt/ltsp/i386/** vom Server exportiert. Das /linuxrc−Skript kann das neue Dateisystem nicht sofort als / mounten. Zuerst muss es unter **/mnt** eingehängt werden. Hiernach führt das Skript das Kommando **pivot_root** aus; pivot_root tauscht das aktuelle root−Dateisystem mit dem neuen aus. War die Aktion erfolgreich, ist das NFS−Dateisystem unter / gemountet und das alte root−Dateisystem unter /oldroot.

10. Nach Mounten und Tauschen des neuen root−Dateisystems ist das /linuxrc−Script abgearbeitet und das eigentliche **/sbin/init**−Programm kann gestartet werden.

11. Der init−Prozess liest die Datei `/etc/inittab` und setzt die Systemumgebung der Workstation entsprechend.

12. Einer der ersten Einträge in der inittab−Datei ist das **rc.sysinit**−Kommando, welches aufgeführt wird, während sich die Workstation in der '***sysinit***'−Phase befindet.

13. Das rc.sysinit−Skript legt eine RAM−Disk von 1 Mb an. Diese wird alles enthalten, was geschrieben oder in irgendeiner Weise verändert werden muss.

14. Die RAM−Disk wird als Verzeichnis `/tmp` gemountet. Alle Dateien, die Schreibrechte besitzen müssen, existieren tatsächlich nur im Verzeichnis `/tmp`. Symbolische Links verweisen dann von der notwendigen Position aus auf diese.

15. Dann wird das `/proc`−Dateisystem gemountet.

16. Anschließend wird die `lts.conf`−Datei analysiert und alle zu dieser Workstation gehörenden Parameter in dieser Datei werden als Environment−Variablen, welche durch das rc.sysinit−Skript genutzt werden, gesetzt.

17. Wurde die Workstation für Swapping über NFS konfiguriert, wird das Verzeichnis **/var/opt/ltsp/swapfiles** unter /tmp/swapfiles gemountet. Dann wird, falls noch keine Swap−Datei für diese Workstation angelegt ist, eine erstellt. Die Größe der Swap−Datei wird in der Datei `lts.conf` festgelegt.

    Die erstellte Swap−Datei wird über das Kommando **swapon** aktiviert.

18. Das Netzwerk−Device ***loopback*** wird konfiguriert. Dies ist dasjenige Netzwerkinterface, welches die IP−Adresse *127.0.0.1* besitzt.

19. Wurde die Workstation so eingerichtet, dass lokale Anwendungen ausgeführt werden können, dann wird nun das **/home**−Verzeichnis vom Server gemountet, so dass die Programme Zugriff auf das Home−Verzeichnis der Benutzer haben.

20. Mehrere Verzeichnisse werden nun im `/tmp/`−Dateisystem angelegt, damit dort Dateien abgelegt werden können, die während der Laufzeit des Client−Rechners notwendig sind. Folgende

Verzeichnisse werden erstellt:

a. `/tmp/compiled`

b. `/tmp/var`

c. `/tmp/var/run`

d. `/tmp/var/log`

e. `/tmp/var/lock`

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

21. Die Datei `/tmp/syslog.conf` wird erstellt. Sie enthält Informationen darüber, zu welchem Host im Netzwerk der **syslog**−Daemon die Logging−Informationen senden soll. Der syslog−Host wird in der Datei `lts.conf` festgelegt. Es gibt einen symbolischen Link namens `/etc/syslog.conf`, der auf die Datei `/tmp/syslog.conf` verweist.

22. Nun kann der **syslogd**−Daemon, für den im vorigen Schritt die Konfigurationsdatei erstellt wurde, gestartet werden.

23. Sobald das rc.sysinit−Skript abgearbeitet ist, wird die Kontrolle an das /sbin/init−Programm zurück gegeben, das dann den Runlevel von ***sysinit*** auf ***5*** ändert.

    Das führt dazu, dass die anderen Einträge in der Datei `/etc/inittab` abgearbeitet werden.

24. Standartmäßig gibt es Einträge in der inittab, welche das Skript **/etc/screen_session** auf tty1, tty2 und tty3 ausführt. Das bedeutet, dass drei Sessions zur gleichen Zeit laufen können. Der Sessiontyp wird durch die Einträge ***SCREEN_01***, ***SCREEN_02*** und ***SCREEN_03*** in der `lts.conf`−Datei bestimmt.

    Falls gewünscht, kann die Datei `inittab` um mehr Einträge für weitere Sessions ergänzt werden.

25. Ist ***SCREEN_01*** auf den Wert ***startx*** gesetzt, wird das Skript **/etc/screen.d/startx** ausgeführt, welches das X−Window−System startet und eine graphische Benutzeroberfläche zur Verfügung stellt.

    In der Datei **lts.conf** gibt es einen Parameter namens **XSERVER**. Fehlt dieser oder ist er auf "**auto**" gesetzt, erfolgt ein Versuch, die Grafikkarte automatisch zu erkennen. Handelt es sich um eine PCI− oder AGP−Grafikkarte, wird der PCI−Vendor und die Device−ID bestimmt und die Datei **/etc/vidlist** wird daraufhin auf einen passenden Eintrag hin überprüft.

    Wird die Karte von Xorg 6.7 unterstützt, gibt die pci_scan−Routine den Namen des Treibermoduls zurück. Wird sie allerdings nur von XFree86 3.3.6 unterstützt, gibt diese Routine den jeweiligen Namen des zu benutzenden XServers zurück. Das startx−Skript kann das unterscheiden, da die alten 3.3.6−Servernamen mit 'XF86_' beginnen, während die neueren X−Server−Module von Xorg typischerweise klein geschrieben werden wie z.B. *ati* oder *trident*.

26. Falls Xorg benutzt wird, erstellt das Skript **/etc/build_x4_cfg** eine XF86Config−Datei. Bei XFree86 3.3.6 wird stattdessen **/etc/build_x3_cfg** aufgerufen. Diese Dateien wrden im Verzeichnis `/tmp` abgelegt, das ja bekanntlich eine RAM−Disk ist und ausschließlich von der jeweiligen Workstation

gesehen wird.

Die XF86Config−Datei wird unter Verwendung der in der Datei **/etc/lts.conf** gemachten Einträge erstellt.

27. Sobald die XF86Config−Datei erstellt ist, wird der X−Server durch das **startx**−Skript unter Verwendung dieser neu erstellten Konfigurationsdatei gestartet.

28. Der X−Server sendet eine ***XDMCP***−Anfrage an den LTSP−Server, der daraufhin einen Login−Dialog zur Verfügung stellt.

29. An diesem Punkt kann sich der Benutzer einloggen und erhält eine Session auf dem Server.

    Anfangs irritiert dies viele Benutzer. Sie sitzen an einer Workstation, aber die Session läuft auf dem Server. Alle von ihnen ausgeführten Kommandos laufen auf dem Server, aber ihre Ausgaben erfolgen auf der Workstation.

1.2. Laden des Kernels in den Speicher

Das Laden des Kernels in den Arbeitsspeicher der Workstation kann auf verschiedene Weisen bewerkstelligt werden.

- Boot ROM
- Lokales Speichermedium

1.2.1. Boot ROM

- Etherboot

  Etherboot ist ein sehr populäres Open−Source−Bootrom−Projekt. Es beinhaltet Treiber für viele gebräuchliche Netzwerkkarten und arbeitet überdies sehr gut mit LTSP zusammen.

  Linux−Kernel müssen mit dem Kommando **mknbi−linux** bearbeitet werden, welches den Kernel für das Booten über das Netzwerk vorbereitet, indem es dem Kernel zusätzlichen Code voranstellt und das initrd−Image an dessen Ende anhängt.

  Die Kernel, die mit LTSP mitgeliefert werden, sind schon bearbeitet und vorbereitet für das Booten mit Etherboot.

  Etherboot kann auch auf eine Diskette geschrieben werden, was beim Testen überaus hilfreich sein kann.

- PXE

  Ein Teil der 'Wired for Management'−Spezifikation aus den späten 90er Jahren enthielt eine Spezifikation für eine Bootrom−Technologie, welche *Pre−boot Execution Environment* genannt wird und üblicherweise mit ***PXE*** abgekürzt wird.

  Ein PXE−Bootrom kann höchstens eine 32 Kilobyte große Datei laden. Ein Linux−Kernel ist ein ganzes Stück größer. Deswegen veranlassen wir PXE dazu, einen Second−Stage−Bootloader −

pxelinux genannt − zu laden. pxelinux ist klein genug, um geladen zu werden und weiß, wie größere Dateien wie ein Linux−Kernel geladen werden können.

- MBA

  Ein Bootrom der Firma ***emBoot*** ist der Managed Boot Agent (MBA). emBoot war früher die Lanworks−Abteilung von 3Com. MBA ist eigentlich eine Sammlung von vier Bootroms. Es kann mit PXE, TCP/IP, RPL und Netware umgehen.

  Die MBA−Implementation von PXE läuft sehr gut. Man kann sie zusammen mit pxelinux zum Booten des Linux−Kernels verwenden.

  Die ***TCP/IP***−Methode kann auch genutzt werden. Allerdings muss der Kernel dafür mit dem Tool **imggen** aufbereitet werden.

- Netboot

  Netboot ist wie Etherboot ein freies Software−Projekt, das frei verfügbare Boot−ROM−Images zur Verfügung stellt. Im Unterschied zu Etherboot ist es ein Wrapper um den NDIS− oder Packet−Treiber, welcher mit der Netzwerkkarte geliefert wird.

1.2.2. Lokale Speichermedien

- Diskette

  Es gibt zwei Wege, um eine LTSP−Workstation über Diskette zu booten. Der eine Weg ist, ein Etherboot−Image in den Bootsektor der Diskette zu laden. Auf diese Weise verhält sie sich wie ein Bootrom. Der Bootcode wird ausgeführt, die Netzwerkkarte wird initialisiert und der Kernel wird vom Netzwerk−Server geladen.

  Man kann auch das initrd−Image und den Kernel auf die Diskette kopieren und auf diese Weise booten. Allerdings ist es schneller, den Kernel über das Netzwerk zu laden.

- Festplatte

  Die Festplatte kann mit LILO oder GRUB genutzt werden, um das initrd−Image und den Kernel zu laden. Oder man kann das Etherboot−Bootrom−Image von der Festplatte laden, was sich dann wie ein Bootrom verhält.

- CD−ROM

  Eine bootfähige CD−ROM kann entweder mit einem Linux−Kernel oder einem Etherboot−Image geladen werden.

- USB−Speichergerät

  Ein USB−Speichergerät kann genau wie eine CD−ROM, eine Diskette oder Festplatte dazu genutzt werden, um entweder ein Etherboot−Modul oder einen kompletten Linux−Kernel und ein initrd−Image zu booten.

Kapitel 2. Installation von LTSP auf dem Server

LTSP kann man sich am besten als eine vollständige Linux−Distribution vorstellen. Sie setzt auf einer Host−Distribution auf. Diese kann jede beliebige Linux−Distribution sein. Tatsächlich braucht das Betriebssystem des Hosts nicht Linux zu sein. Die einzige Anforderung ist, dass das Host−System als NFS−Server (NFS = Network File System) fungieren kann. Die meisten UNIX−Systeme können diese Bedingung erfüllen. Tatsächlich gibt es sogar einige Versionen von Microsoft Windows, die als LTSP−Server konfiguriert werden können.

Es müssen drei Phasen beim Aufbauen eines LTSP−Servers durchlaufen werden:

- Installieren der LTSP−Hilfsprogamme
- Installieren der LTSP−Client−Pakete
- Konfigurieren der von LTSP benötigten Dienste

2.1. Installation der LTSP−Hilfsprogramme

Seit der Version 4.1 besitzt LTSP ein Paket von Hilfsprogrammen, um die LTSP−Client−Pakete (die Programme, die auf den Thin Clients laufen) zu installieren und zu verwalten und um die Dienste auf dem LTSP−Server zu konfigurieren.

Das Adimistrations−Utility heißt **ltsadmin**, das Konfigurations−Tool **ltspcfg**. Diese beiden Hilfsprogramme sind Teil des **ltsp−utils**−Pakets.

Das ***ltsp−utils***−Paket ist sowohl im ***RPM***− als auch im ***TGZ***−Format verfügbar. Wählen Sie Ihr bevorzugtes Format aus und folgen Sie den jeweiligen Instruktionen.

2.1.1. Installation des RPM−Pakets

Laden Sie die neueste Version des ltsp−utils RPM−Pakets herunter und installieren Sie es unter Verwendung des folgenden Kommandos:

```
rpm −ivh ltsp−utils−0.11−0.noarch−rpm
```

Dieses Kommando installiert die Hilfsprogramme auf dem Server.

2.1.2. Installation der TGZ−Pakete

Laden Sie die neueste Version des ltsp−utils TGZ−Pakets herunter und installieren Sie es unter Verwendung der folgenden Befehle:

```
tar xzf ltsp−utils−0.11−0.noarch.tgz
cd ltsp_utils
./install.sh
cd ..
```

Mit Hilfe dieser Befehle werden die Hilfsprogramme auf dem Server installiert. Diese Installationsmethode sollte auf nicht−RPM−basierten Systemen angewandt werden.

2.2. Installation der LTSP−Client−Pakete

Sobald die Installation des ltsp−utils−Pakets abgeschlossen ist, kann das Kommando **ltspadmin** aufgerufen werden. Dieses Hilfsprogramm kann zur Verwaltung der LTSP−Client−Pakete genutzt werden. Es schickt eine Anfrage an das LTSP−Download−Repository und bekommt so eine Liste der zurzeit verfügbaren Pakete.

Nach dem Aufruf des **ltspadmin**−Kommandos erscheint ein Bildschirm, der in etwa folgendes zeigt:

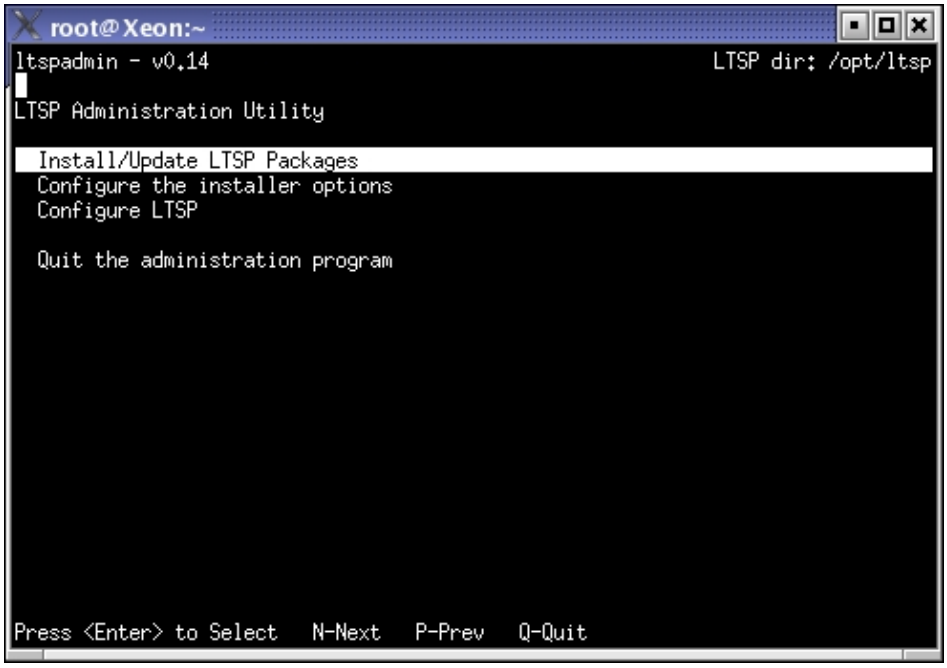

Abbildung 2−1. Der Hauptbildschirm des LTSP−Installers

Aus dem Menü können Sie "Install/Update" auswählen; wenn Sie das erste Mal dieses Hilfsprogramm aufgerufen haben, wird der Installer−Konfigurations−Bildschirm angezeigt.

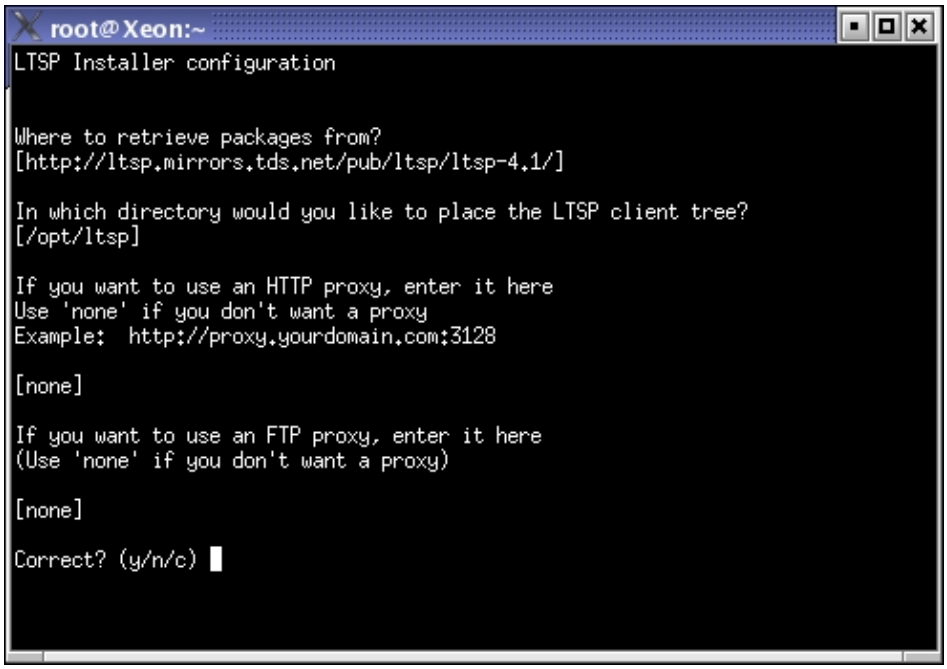

Abbildung 2−2. Der Konfigurationsbildschirm des LTSP−Installers

In diesem ***Konfigurations***−Bildschirm können mehrere Werte gesetzt werden, die der Installer für das Herunterladen und Konfigurieren der Pakete benötigt. Diese Werte sind:

Where to retrieve the packages from? *(Von welchem Server sollen die Pakete bezogen werden?)*
: Das ist eine URL, die auf das Paket−Repository verweist. Typischerweise wird es `http://www.ltsp.org/ltsp-4.1` sein. Wenn Sie aber die Pakete von einem lokalen Dateisystem aus installieren wollen, können Sie `file:` stattdessen verwenden. Befinden sich die Pakete z.B. auf einer CD−ROM, die unter `/mnt/cdrom` gemountet ist, muss der Wert für das Paket−Respository wie folgt aussehen: `file:///mnt/cdrom`. (Beachten Sie bitte die drei /−Zeichen!)

In which directory would you like to place the LTSP client tree? *(In welchem Verzeichnis soll das LTSP−Client−Verzeichnis angelegt werden?)*
: Dies ist das Verzeichnis auf dem Server, wo Sie den LTSP−Client−Verzeichnisbaum anlegen wollen. Typischerweise sollte es `/opt/ltsp` sein. Dieses Verzeichnis wird erstellt, sofern es noch nicht existiert.

  In diesem Verzeichnis werden die root−Verzeichnisse angelegt, pro Systemarchitektur eins. Zurzeit werden nur x86−Workstations offiziell unterstützt (i386−Architektur). Allerdings gibt es viele Leute, die an Portierungen auf andere Architekturen wie PPC und Sparc arbeiten.

HTTP Proxy
: Befindet sich der Server hinter einer Firewall und der Zugriff auf das Internet erfolgt immer über einen Proxy−Server, können Sie den Installer so einstellen, dass er diesen Proxy verwendet. Der Wert für den Proxyserver sollte dessen URL, das verwendete Protokoll und den Port enthalten. Ein Beispiel für solch eine Einstellung ist: `http://firewall.yourdomain.com:3128`

  Benötigen Sie keinen Proxy, sollten Sie "`none`" eintragen.

FTP Proxy

Liegen die Pakete auf einem FTP−Server und Sie müssen, um auf diesen zugreifen zu können, über einen FTP−Proxy gehen, geben Sie diesen hier an. Die Syntax ist analog zu der des HTTP−Proxys von oben.

Benötigen Sie keinen Proxy, sollten Sie auch hier "none" eintragen.

Sobald Sie mit den Eingaben auf dem Konfigurationsbildschirm fertig sind, fragt der Installer das Paket−Repository ab und erhält eine Liste der zurzeit verfüg− und damit installierbaren Komponenten.

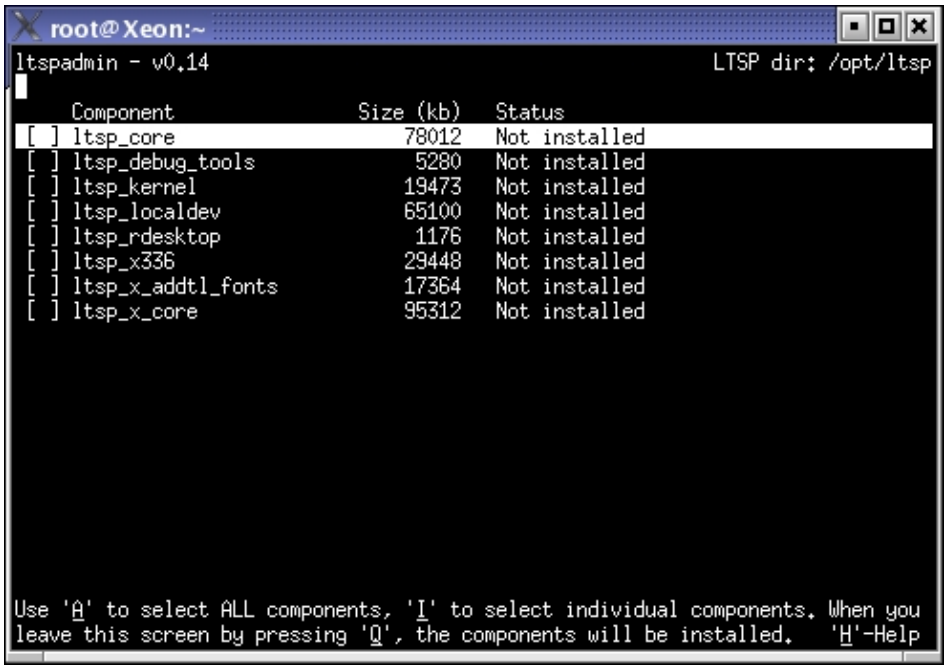

Abbildung 2−3. Die vom LTSP−Installer erhaltene Komponentenliste

Wählen Sie die Komponente aus, die Sie installieren möchten. Bewegen Sie dazu die hervorgehobene Zeile auf die von Ihnen gewünschte Komponente und wählen Sie sie durch einen Druck auf die Taste '**I**' aus. Sie können auch '**A**' drücken, um alle Komponenten auszuwählen. Meistens ist es das, was Sie wollen. Auf diese Weise wird eine große Palette von Thin−Client−Hardware unterstützt.

Es gibt noch ein paar mehr Tasten, die Sie in diesem Bildschirm verwenden können. Durch Drücken der '**H**'−Taste erhalten Sie ein Hilfemenü, in dem diese Tasten genannt werden.

Abbildung 2−4. Das Hilfefenster des LTSP−Installers

Wollen Sie eine Liste der Pakete sehen, welche zu einer bestimmten Komponente gehören, drücken Sie die Taste '**S**'. Nun werden die Pakete angezeigt. Außerdem erhält man Informationen über die zurzeit installierten Pakete und deren aktuell verfügbare Versionen.

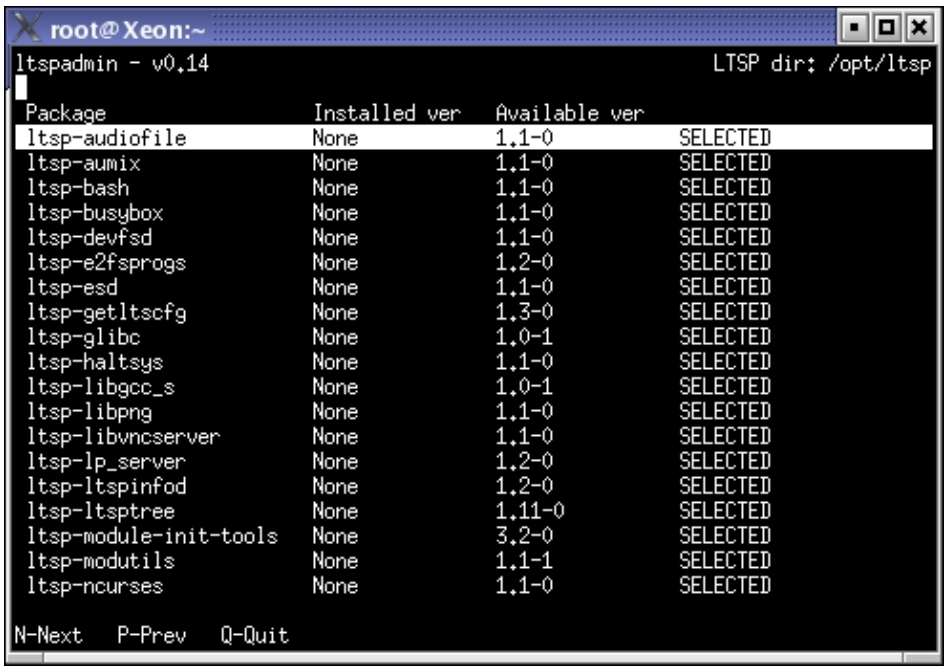

Abbildung 2−5. Die Paketliste im LTSP−Installer

Sobald Sie alle gewünschten Komponenten ausgewählt haben, können Sie den Komponentenauswahl−Bildschirmm verlassen. Der Installer wird Sie dann fragen, ob Sie wirklich die

gewählten Pakete updaten/installieren wollen. Bestätigen Sie das mit der '**Y**'−Taste, wird der Installer mit dem Herunterladen und dem Installieren der Pakete beginnen.

2.3. Konfiguration der von LTSP benötigten Dienste

Es gibt vier grundlegende Dienste, die für das Booten einer LTSP−Workstation benötigt werden. Diese sind:

- DHCP
- TFTP
- NFS
- XDMCP

Das Kommando **ltspcfg** kann dafür verwendet werden, alle diese Dienste zu konfigurieren plus einiger LTSP−bezogenen Dinge mehr.

Sie können **ltspcfg** über das Programm **ltspadmin** oder direkt über die Kommandozeile aufrufen, indem Sie **ltspcfg** eingeben.

Wenn Sie das ltspcfg−Hilfprogramm ausführen, scannt es den Server, um zu schauen, was installiert ist und was gerade ausgeführt wird. Sie bekommen einen Bildschirm, der in etwa folgendermaßen aussieht:

Abbildung 2−6. Der Eingangsbildschirm von ltspcfg

Dieser Bilschirm zeigt alle durch den Scanprozess des Hilfsprogramms erhaltenen Informationen an.

Durch Drücken der Taste '**C**' gelangen Sie zu dem Konfigurationsmenü. Sie müssen jeden einzelnen Eintrag durchgehen, um sicherzustellen, dass der LTSP−Server für die Bedienung der Workstations korrekt konfiguriert ist.

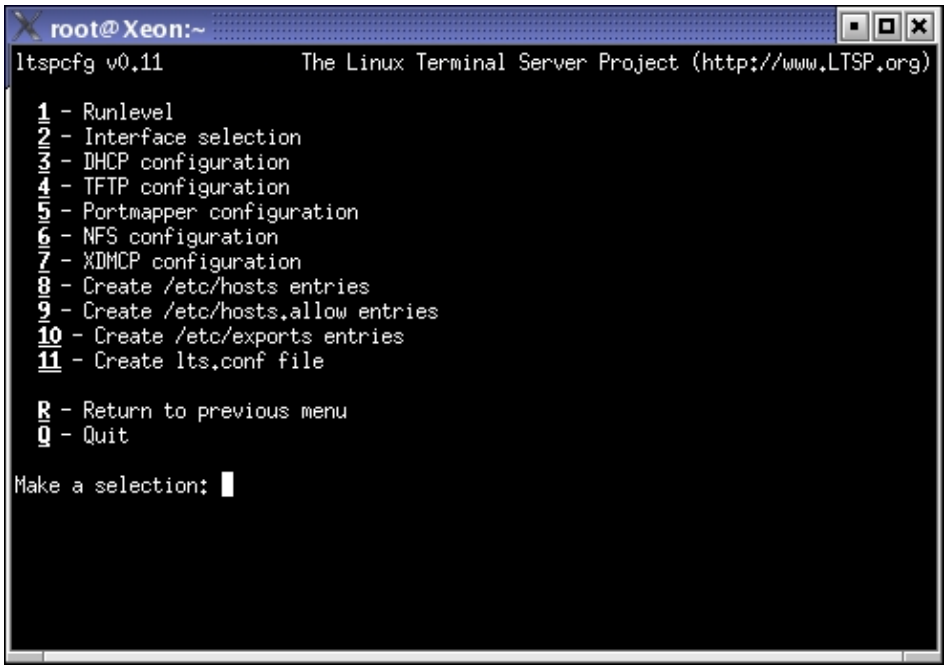

Abbildung 2−7. Das ltspcfg−Konfigurationsmenü

1 − Runlevel

Die Variable **Runlevel** wird durch das Programm **init** verwendet. Bei Linux− und Unix−Sytemen befindet sich das System zu jedem Zeitpunkt in einem spezifischen "Runlevel". Die Runlevel 2 und 3 werden typischerweise genutzt, wenn sich der Server im Text−Modus befindet. Runlevel 5 hingegen zeigt typischerweise an, dass sich das System im graphischen Modus in einem Netzwerk befindet.

Bei einem LTSP−Server wird traditionell der Runlevel 5 verwendet. Die meisten Systeme sind bereits so konfiguriert, dass sie NFS und XDMCP in jenem Runlevel bedienen können. Dieses Hilfsprogramm konfiguriert diejenigen Systeme, die dies nicht können, entsprechend.

2− Interface selection

Sollte das System über mehrere Netzwerkkarten verfügen, können Sie hier dasjenige Interface auswählen, mit welchem die Thin Clients verbunden sind.

Durch die Auswahl des Interfaces kann das Konfigurationstool korrekte Konfigurationsdateien wie z.B. die Datei `dhcpd.conf` oder die `/etc/exports`−Datei erstellen.

3 − DHCP configuration

DHCP muss so konfiguriert werden, dass die von der Workstation benötigten Felder an sie weitergegeben werden. Unter diesen Feldern sind die folgenden: **fixed−address**, **filename**, **subnet−mask**, **broadcast−address** und **root−path**.

Durch Auswahl dieses Menüpunkts können Sie eine `dhcpd.conf`−Konfigurationsdatei erstellen und den **dhcpd**−Daemon so konfigurieren, dass er beim Systemstart automatisch geladen wird.

4 − TFTP configuration

TFTP wird vom Thin Client genutzt, um den Linux−Kernel herunterzuladen. Der **tftpd**−Dienst muss auf dem Server aktiviert werden, um den Kernel zum Download bereit stellen zu können.

5 − Portmapper configuration

Der **Portmapper** wird von den RPC−Diensten genutzt. Jeder RPC−Service wie z.B. NFS muss sich beim Portmapper beim Start registrieren und ihm u.a. die von ihm verwendete Portnummer mitteilen. Bei Anfragen eines RPC−Clients an den Portmapper nach einem konkreten RPC−Dienst durchsucht dieser seine Datenbasis und übergibt, falls der Dienst verfügbar ist, dessen Portnummer. Alle Clientzugriffe erfolgen später direkt an diesen Port.

6 − NFS configuration

Der NFS−Dienst erlaubt es, dass lokale Verzeichnisbäume auf entfernten Maschinen gemountet werden können. Diese Eigenschaft ist für LTSP erforderlich, da die Workstations ihr root−Dateisystem vom Server mounten.

Unter diesem Menüpunkt wird der NFS−Daemon so eingerichtet, dass er beim Bootvorgang automatisch gestartet wird. Die Konfigurationsdatei `/etc/exports` und ihre Erstellung werden später in diesem Abschnitt beschrieben.

7 − XDMCP configuration

XDMCP heißt: "X Display Manager Control Protocol". Der X−Server sendet eine XDMCP−Anfrage an den Display−Manager des Servers, um einen Login−Prompt zu erhalten.

Gewöhnlich werden **XDM**, **GDM** und **KDM** als Display−Manager verwendet. Dieser Menüpunkt zeigt die gefundenen Display−Manager an. Außerdem erfolgt naoch die Anzeige des Display−Managers, der für die Ausführung konfiguriert ist.

Aus Sicherheitsgründen ist der Display−Manager so konfiguriert, dass sich Remote−Workstations mit ihm verbinden können. Dies ist die Ursache für den berühmten ***grauen Bildschirm mit dem großen X−Cursor***. Gewöhnlich kann ltspcfg den Display−Manager so konfigurieren, dass Remote−Workstations eine Verbindung mit ihm bekommen.

8 − Create /etc/host entries

Viele Dienste wie NFS und der Display−Manager müssen in der Lage sein, der IP−Adresse einer Workstation einen entsprechenden Hostnamen zuzuordnen. Dies kann durch den Einsatz des Berkeley Internet Naming Daemon (BIND) erreicht werden. Allerdings muss dabei sicher gestellt sein, dass auch der ***Reverse***−Lookup richtig funktioniert (d.h. einem Hostnamen muss die entsprechende IP−Adresse zugeordnet werden können). Der Einsatz von BIND ist letztlich die beste Wahl, um dies zu erledigen, aber die Konfiguration von BIND würde den Rahmen dieses Dokuments sprengen und hat nichts mit dem ltspcfg−Hilfsprogramm zu tun.

Eine einfachere Methode, um IP−Adressen Hostnamen zuzuordnen, ist die Verwendung der Datei `/etc/hosts`.

9 − Create /etc/hosts.allow entries

Einige Dienste benutzen einen Security−Layer, welcher ***tcpwrappers*** genannt wird. Dieser wird durch die Datei `/etc/hosts.allow` konfiguriert. Dieser Menüpunkt erstellt eine solche Datei für Sie.

10 − Create the /etc/exports file

Anfhand dieser Datei kann NFS feststellen, welche Verzeichnisse auf entfernten Maschinen gemountet werden dürfen. Dieser Menüpunkt erstellt diese Datei.

11 − Create the lts.conf file

Die Konfiguration jeder einzelnen Workstation erfolgt anhand der Einträge in der Datei `lts.conf`. Werden relativ neue Workstations mit PCI−Bus eingesetzt, braucht die Datei lts.conf keine zusätzlichen Einträge. Sie muss allerdings existieren. Dieser Menüpunkt erstellt die Default−lts.conf−Datei für Sie.

2.4. Workstation−spezifische Konfiguraton

Nun ist es an der Zeit, den LTSP−Server über jede einzelne Workstation in Kenntnis zu setzen. Es gibt drei Dateien, die Informationen über diese Workstation enthalten.

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

2.4.1. /etc/dhcpd.conf

Die Workstation braucht eine IP−Adresse und weitere Informationen. Sie bekommt folgende Informationen vom DHCP−Server:

- IP−Adresse
- Hostname
- Server−IP−Adresse
- Default Gateway
- Pfadname des zu ladenden Kernels
- Servername und Verzeichnispfad auf dem Server, der als root−Dateisystem gemountet werden soll

In unserer Beispielkonfiguration weisen wir per DHCP−Server den Clients ihre IP−Adressen zu.

Während das ltspcfg−Skript abläuft, wird eine Beispieldatei für dhcpd.conf erstellt. Sie heißt `/etc/dhcpd.conf.example`. Sie können sie nach /etc/dhcpd.conf kopieren und diese Datei dann als Basis für Ihre DHCP−Konfiguration benutzen. Selbstverständlich müssen Sie Anpassungen an Ihre eigene Systemumgebung vornehmen und die Angaben zu den Workstations anpassen.

```
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";
    }
}
```

Abbildung 2−8. /etc/dhcpd.conf

Seit der Version 2.09pre2 von LTSP ist es nicht mehr notwendig, einen speziellen Kernel anzugeben. Die Kernel des Standard−Pakets enthalten Treibermodule für alle von Linux unterstützten Netzwerkkarten. Es gibt zwei Kernel im LTPS−Kernel−Paket. Einer enthält den Linux Progress Patch (LPP), der andere nicht. Sie sind an ihren Namen zu erkennen; beispielsweise:

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

Vielleicht ist Ihnen aufgefallen, dass der Kernel im Verzeichnis `/tftpboot/lts` liegt, der Eintrag "filename" in der Datei `/etc/dhcpd.conf` aber nicht den Vorsatz `/tftpboot` enthält. Das hat folgenden Grund: Seit der Red−Hat Version 7.1 läuft TFTP mit der Option '−s', d.h. der Prozess tftpd läuft im ***secure***−Modus: Beim Start wird ein **chroot** zum Verzeichnis `/tftpboot` ausgeführt. Damit sind für den tftpd−Prozess alle Dateien relativ zum Verzeichnis `/tftpboot` erreichbar.

Andere Linux−Distribution verwenden möglicherweise die Option '−s' nicht für tftpd. Daher müssen Sie `/tftpboot` vor den Pfadnamen des Kernels setzen.

2.4.2. /etc/hosts

Umwandlung von IP−Adressen zu Hostnamen

Computer kommen i.a. mit IP−Adressen aus. Menschen haben lieber Namen für ihre Rechner. Für die Zuordnung gibt es Nameserver (DNS) oder die Datei `/etc/hosts`. In einer LTSP−Umgebung ist eine Umwandlung von IP−Adressen in Hostnamen unverzichtbar, da es sonst keinen Zugriff der Workstations auf das vorgesehene root−Verzeichnis auf dem Server per NFS gibt.

Außerdem könnte es bei fehlendem Eintrag der Workstation in der Datei `/etc/hosts` zu Problemen mit den Display−Managern ***GDM*** oder ***KDM*** kommen.

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

In der Datei `lts.conf` gibt es eine ganze Reihe von Konfigurationseinträgen.

Die Datei `lts.conf` hat eine einfache Syntax und besteht aus mehreren Sektionen. Es gibt eine Sektion, die für alle Workstations gilt: **[default]** und es kann für einzelne Workstations spezifische Einträge geben. Die Workstation kann durch den Rechnernamen, die IP−Adresse oder die MAC−Adresse identifiziert werden.

Eine `lts.conf`−Datei sieht üblicherweise folgendermaßen aus:

```
#
# 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
```

Beispiel 2−1. lts.conf−Datei

Es folgt eine Liste einiger Konfigurationsvariablen:

XSERVER

Ist Ihre Grafikkarte eine PCI−Karte und wird diese von X.org 6.7.0 unterstützt, dann brauchen Sie nur das lts_x_core−Paket. Dieses enthält alle für X4 notwendigen Treibermodule.

Es gibt auch mehrere XFree86−3.3.6−Pakete für LTSP für den Fall, dass Ihre Karte nicht von X.org 6.7.0 unterstützt wird.

Einträge können in der Datei `lts.conf` pro Workstation oder zentral in der default−Sektion erfolgen.

Da unsere Beispiel−Workstation über einen Intel i810 Video−Chipsatz verfügt, der automatisch erkannt wird, braucht kein XSERVER−Eintrag gemacht zu werden. Wenn Sie wollen, können Sie 'auto' oder das entsprechende Treibermodul eintragen.

RUNLEVEL

Wir wollen die Workstation im Grafikmodus betreiben, daher setzen wir den Runlevel auf '5'. Dies wird durch einen weiteren Eintrag in der Datei `lts.conf` gemacht.

2.5. Anzeigen der aktuellen Konfiguration

Mit **ltspcfg** können Sie einen Überblick über den aktuellen Status der für LTSP erforderlichen Dienste erhalten. Dazu müssen Sie im ltspcfg−Hauptmenü die Taste '**S**' drücken, um den aktuellen Status angezeigt zu bekommen.

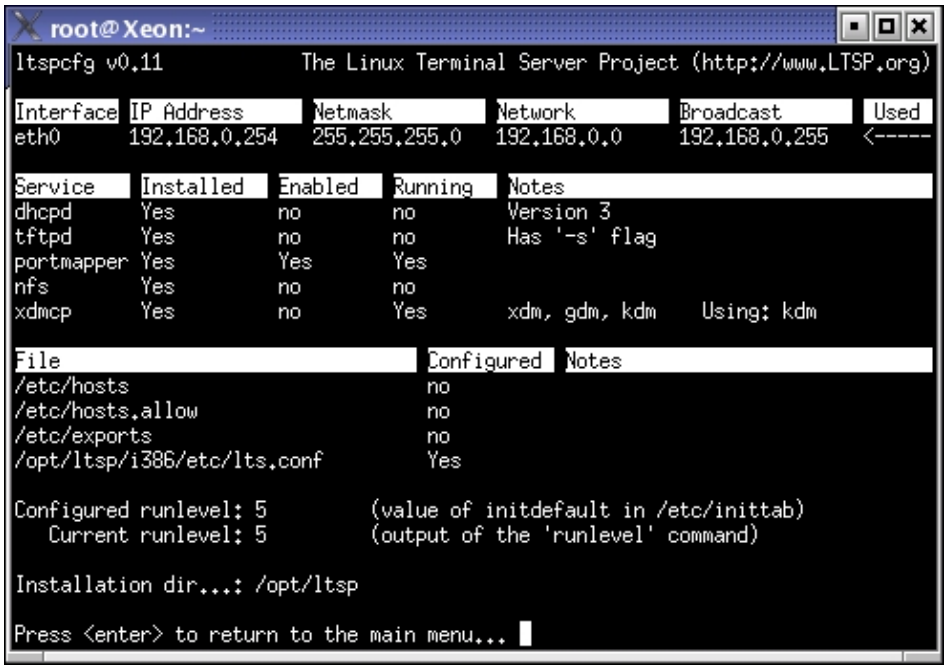

Abbildung 2−9. ltspcfg − Aktueller Status

Kapitel 3. Einrichtung der Workstation

Nachdem die Einrichtung des Servers abgeschlossen ist, ist es an der Zeit, den Fokus auf die Workstation zu verlagern.

LTSP beginnt dann, wenn der Kernel in den Arbeitsspeicher der Workstation geladen wurde. Es gibt mehrere Wege, den Kernel zu laden: Etherboot, Netboot, PXE und Diskette.

3.1. Booten mit PXE

Verfügt Ihre Netzwerkkarte oder Ihr PC über PXE, kann es zum Laden des Linux−Kernels verwendet werden. PXE ist eine Bootrom−Technologie, die der von Ether− oder Netboot ähnlich ist.

Sie müssen möglicherweise das PXE−Bootrom auf Ihrer Netwerkkarte aktivieren. Außerdem müssen Sie eventuell die Bootreihenfolge in Ihrem BIOS ändern. Sie muss so eingestellt werden, dass "Boot from LAN" der erste Eintrag in der Bootreihenfolge ist und nicht das Diskettenlaufwerk oder gar die Festplatte.

PXE besitzt eine Einschränkung: es ist nur in der Lage, Dateien zu laden, die 32kb oder kleiner sind. Der Linux−Kernel ist wesentlich größer. Sie können ihn also nicht direkt mit PXE laden. Sie brauchen hierfür ein Programm, was als 'Network Bootstrap Program' oder kurz als NBP bezeichnet wird.

Zum Laden des Linux−Kernels gibt es ein NBP namens **pxelinux.0**. Es ist Bestandteil des Pakets **syslinux** von Kernel−Entwickler H. Peter Anvin.

Das LTSP−Kernel−Paket enthält das pxelinux.0−NBP und eine Konfigurationsdatei, mit deren Hilfe der Linux−Kernel und ein Initial−RAMdisk−Image geladen werden können.

Der Ladevorgang läuft folgendermaßen ab:

- Das PXE−Bootrom initialisiert die Netzwerkarte und sendet eine DHCP−Anfrage.
- Der DHCP−Server antwortet, indem er eine IP−Adresse und den Namen des zu ladenden NBPs zurück schickt.
- Das PXE−Bootrom lädt das NBP herunter, platziert es im Speicher und beginnt mit dessen Ausführung.
- Das NBP nutzt tftp, um die Konfigurationsdatei vom Server herunter zu laden.
- Die Konfigurationsdatei enthält den Namen des Kernels, den Namen der Initial−RAMdisk−Datei und Optionen, welche dem Kernel übergeben werden nachdem dieser geladen ist.
- Eine pxelinux−Konfigurationsdatei sieht in etwa wie folgt aus:

```
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
```

- Das NBP benutzt nun tftp, um den Linux−Kernel und die Initial−RAMdisk−Datei (initrd) herunter zu laden.
- Die Kontrolle wird dann an den Linux−Kernel gegeben. Dieser bootet, mountet das initrd−Image und setzt dann das Starten des Thin Clients fort.

3.2. Booten mit Etherboot

> Etherboot ist ein Softwarepaket zum Erstellen von ROM−Images, die über das Netzwerk geladen werden und auf einem x86−Computer ausgeführt werden können. Viele Netzwerkkarten besitzen einen Sockel, auf den ein ROM−Chip gesteckt werden kann. Etherboot ist der Code, der auf ein solches ROM gebracht werden kann.
>
> −−Ken Yap

Etherboot ist auch Open Source und durch die GNU General Public License Version 2 (GPL2) geschützt.

Wenn Sie eine Netzwerkkarte mit Etherboot−Bootrom besitzen und diese zusammen mit Etherboot einsetzen wollen, müssen Sie Ihr BIOS so einstellen, dass "Boot from LAN" der erste Eintrag in der Bootreihenfolge ist und nicht das Diskettenlaufwerk oder gar die Festplatte.

Besitzen Sie kein Etherboot−Bootrom, können Sie entweder ein Bootrom herstellen oder eine Diskette mit einem Etherboot−Image erstellen, das sich dann im Bootsektor der Diskette befindet.

Etherboot unterstützt eine breite Palette an Netzwerkkarten. Derzeit sind es über 200 und es kommen immer neue hinzu. Gleichgültig, ob Sie eine Diskette erstellen oder den Code auf ein Eprom brennen: Sie müssen vorher ermitteln, welches Netzwerkkartenmodell Sie besitzen.

3.2.1. Auswählen eines Etherboot−Treibers für eine ISA−Netzwerkkarte

Bei älteren ISA−basierten Netzwerkkarten ist es nicht so wichtig, dass Sie den exakten Typ bestimmen. Die meisten Karten sind entweder ne2000− oder 3Com 3c509−Karten. Sie müssen lediglich den richtigen Etherboot−Treiber wählen: Es muss derjenige sein, der auf Karten mit zwei Anschlussmöglichkeiten, wie 10Base−2 (Koax) und 10Base−T (Twisted Pair), den korrekten Medientyp auswählt.

3.2.2. Auswählen eines Etherboot−Treibers für eine PCI−Netzwerkkarte

Bei PCI−Karten ist es wichtig, dass Sie den Etherboot−Treiber auswählen, der mit dem PCI−Vendor und der Device−ID Ihrer Netzwerkkarte übereinstimmt.

Im besten Falle wissen Sie die exakte Modellbezeichnung Ihrer Karte, da sie auf der Karte aufgedruckt ist. Wenn diese Bezeichnung ganz genau mit der Bezeichnung eines der Etherboot−Module übereinstimmt, haben Sie Glück. Meistens jedoch müssen Sie die PCI−ID−Nummern erst finden.

Ist Ihre Workstation mit einem Diskettenlaufwerk ausgestattet, können Sie eine tomsrtbt−Diskette (Tom's Root Boot) booten. Besitzt Ihre Workstation ein CD−ROM−Laufwerk, können Sie einfach eine Knoppix−CD booten. Wenn Sie Linux auf Ihrer Workstation allerdings nicht booten können, müssen Sie als letzten Versuch die Netzwerkkarte in einen Rechner einbauen, der in der Lage ist, Linux zu starten.

Nachdem Sie Linux gestartet haben, können Sie das Kommando **lspci** mit der Option '−n' nutzen. Sie erhalten dann in etwa folgende Bildschirmausgabe:

```
[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)
```

In dem obigen Beispiel können Sie sehen, dass ein Eintrag für jede PCI−Karte im System existiert. Sie brauchen sich nur die ***Class 0200***−Geräte anschauen. Sie können das Kommando nochmal aufrufen und sich diesmal nur die Ethernet−Interfaces anzeigen lassen. Auf diese Weise wird die Liste übersichtlicher.

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

Die PCI−ID−Nummern sind: **8086:1229**. Das erste Feld **8086** bezeichnet den PCI−Vendor (Hersteller). In diesem Beispiel ist der Hersteller die Intel Corporation. Das zweite Feld **1229** bezeichnet die PCI−Device−ID. Diese gibt Aufschluss über das Modell der Netzwerkkarte. Hier ist es eine EtherExpress−100−Karte.

3.2.3. Eine Bootdiskette erstellen

Sie können das Etherboot−Paket herunter laden und ein passendes Image herstellen, das dann in ein EPROM gebrannt oder erst einmal zu Testzwecken auf eine Diskette kopiert werden kann.

Es ist eine einfachere Alternative, auf Marty Connors Webseite unter der URL *www.Rom−O−Matic.net* zu gehen.

Marty hat ein exzellentes web−basiertes Frontend geschaffen, um ein angepasstes Bootrom mit Etherboot zu generieren. Man wählt einfach seine Netzwerkkarte aus und gibt an, welche Art von Image es sein soll. Danach haben Sie die Möglichkeit, viele Konfigurationsoptionen zu modifizieren. Nachdem alles eingegeben ist, reicht ein Klick auf den Button 'Get ROM', um das Image generieren zu lassen.

Als 'ROM output format' wählen Sie 'Floppy Bootable ROM Image'. Dadurch bekommt das Image einen 512 Byte großen Header als Boot−Loader für das eigentliche Etherboot−Image verpasst. Der Boot−Loader lädt beim Start des Rechners per Diskette das Image in den Arbeitsspeicher, wo es dann ausgeführt werden kann.

Jetzt ist der 'Get ROM' Button an der Reihe. Es dauert nur einige Sekunden, bis das Image generiert worden ist und auf dem eigenen Rechner gespeichert werden kann. Nach einiger Zeit erscheint ein 'Speichern als'−Fenster und Sie können auswählen, wohin das Bootrom−Image auf Ihrem Rechner gespeichert werden soll.

Nachdem Sie das Image auf der Festplatte gespeichert haben, müssen Sie es auf eine Diskette schreiben. Legen Sie die Diskette in das Laufwerk ein und geben Sie das folgende Kommando ein, um das Image auf die Diskette zu schreiben:

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

3.2.4. Erstellen eines Bootroms

Um ein Etherboot−Image auf ein EPROM zu schreiben, benötigen Sie einen EPROM−Schreiber. Dieses Gerät kann von mehreren hundert bis zu einigen tausend Dollar kosten, was von dem Leistungsumfang abhängig ist.

Der Bootrom−Erstellungsprozess ist komplett vom EPROM−Schreiber abhängig. Die Beschreibung eines solchen Vorgangs würde den Rahmen dieses Dokuments sprengen.

Kapitel 4. Starten der Workstation

Wurde alles richtig auf dem Server konfiguriert, kann die Workstation per Diskette oder Netzwerk (PXE, Etherboot) gestartet werden.

Der Bootcode wird in den Speicher geladen, die Netzwerkkarte wird gefunden und initialisiert, eine DHCP−Anfrage wird ins Netzwerk geschickt, der DHCP−Server antwortet, der Kernel wird vom Server heruntergeladen und gestartet. Der Kernel erkennt die Hardware, X startet und ein graphisches Login erscheint auf dem Monitor, ähnlich wie im unten gezeigten Beispiel.

Abbildung 4−1. Login screen

Ein normales Einloggen sollte nun möglich sein. Wichtig ist: Sie melden sich auf dem Server an und arbeiten dort. Alle Kommandos werden auf dem Server aufgeführt, die Ausgaben werden auf dem Monitor der Workstation dargestellt. Das ist die Netzwerkfähigkeit des X−Window−Systems.

Alle Programme auf dem Server stehen zur Verfügung.

Kapitel 5. Drucken

Die Workstation kann neben ihrer Rolle als voll funktionsfähiges (ASCII− oder graphisches) Terminal auch noch als Printserver genutzt werden. Bis zu drei Drucker können an die parallelen und seriellen Ports angeschlossen werden.

Für den Benutzer der Workstation ist dies transparent; die zusätzliche Netzwerklast wird kaum bemerkbar sein.

5.1. Einrichten der Workstation

LTSP benutzt das **lp_server**−Programm auf der Workstation, um die Druckaufträge des Servers auf ihre Ports umzuleiten.

Es gibt einige Einträge in der Konfigurationsdatei **lts.conf**, um das Drucken über eine Workstation zu steuern:

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

Der obige Eintrag lässt das lp_server−Programm als Daemon laufen, der auf dem TCP/IP−Port 9100 auf Druckaufträge wartet. Der Drucker am ersten parallelen Anschluss (/dev/lp0) der Workstation bekommt die Daten durchgereicht.

Es gibt viele weitere Einstellmöglichkeiten für das Drucken. Sie finden diese weiter unten in diesem Dokument in der Beschreibung von lts.conf.

5.2. Einrichten des Servers

Das Einrichten auf dem Server beinhaltet die Definition einer Druckerwarteschlange mit einem entsprechenden Konfigurationstool.

Fedora Core (Redhat) besitzt dafür sowohl GUI− als auch textbasierte Tools. Das GUI−Tool heißt **printconf−gui** und das textbasierte **printconf−tui**. Außerdem gibt es noch ein Programm namens **printtool**. Printtool gibt es auch in Fedora, aber in diesem Beispiel benutzen wir printconf−gui. Andere Linux−Distributionen haben ihre eigenen Konfigurationstools.

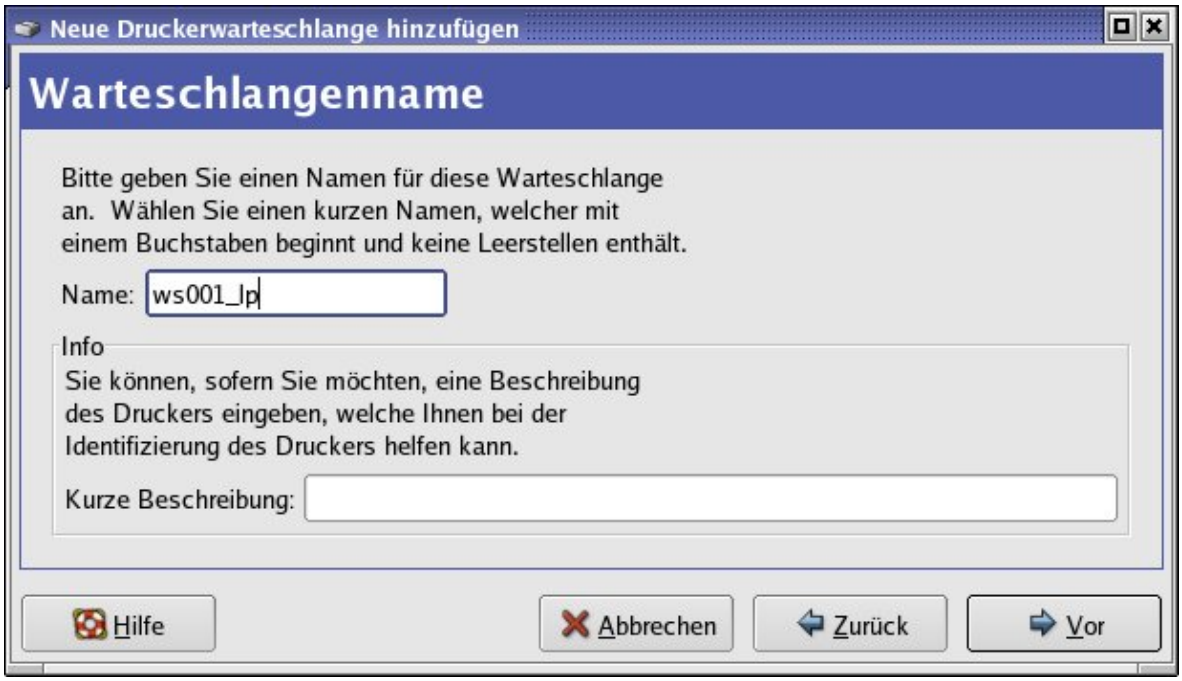

Abbildung 5−1. Einen neuen Drucker mit printconf−gui erstellen

Nach Aufruf des Tools definieren wir einen neuen Drucker. Das lp_server−Programm kann auf der Workstation einen HP−JetDirect−Printserver emulieren. Auf dem Server muss daher ein JetDirect−Drucker eingerichtet werden.

Der neu einzurichtende Drucker braucht einen Namen für die Warteschlange. Dieser Name kann zwar beliebig gewählt werden, sollte aber sinnvoll sein und aus folgenden Zeichen zusammengesetzt:

- `"a−z" Kleinbuchstaben`
- `"A−Z" Großbuchstaben`
- `"0−9" Ziffern`
- `"−" Bindestrich`
- `"_" Unterstrich`

Im Beispiel verwenden wir den Namen **ws001_lp**. Damit ist klar, dass dieser Drucker an der Workstation **ws001** betrieben wird.

Abbildung 5−2. printconf−gui Detail Info

Zwei Felder sind auszufüllen:

1. Die IP−Adresse oder der Name der Workstation, an die der Drucker angeschlossen ist.
2. Der TCP−Port des **lp_server**−Daemons.

   Für den ersten Drucker, der mit der Workstation verbunden ist, ist das TCP/IP−Port **9100**, für weitere sind es die Ports **9101** und **9102**.

Nachdem diese Eingaben getätigt wurden, kann im nächsten Fenster, das Sie durch einen Klick auf '**Vor**' erreichen, der Hersteller und das Modell des Druckers, welcher mit der Workstation verbunden ist, ausgewählt werden.

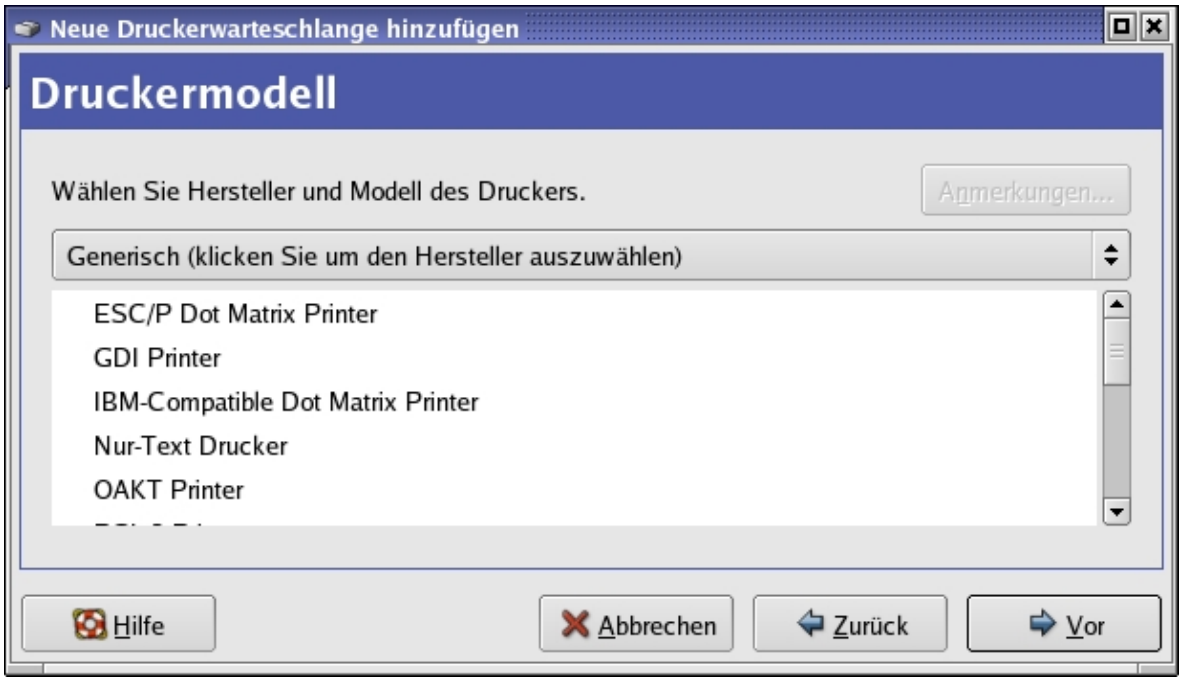

Abbildung 5−3. Die Druckerauswahl im printconf−gui−Programm

Kapitel 6. Screen−Skripte

In der Verson 4.0 wurde LTSP um die sogenannten **Screen−Skripte** ergänzt. Diese Skripte erlauben den Start von verschiedenen Sessiontypen.

Sie können mehrere Screen−Skripte für eine Workstation festlegen und erhalten damit mehrere Sessions. Dies können verschiedene oder gleiche Sessionstypen sein. Sie können beispielsweise folgendes angeben:

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

Auf dem ersten Screen würde hierbei ein X−Server und auf dem zweiten eine Shell−Eingabeaufforderung gestartet. Sie können durch Drücken der Tastenkombination Strg−Alt−F1 auf den ersten Screen wechseln. Wenn Sie Strg−Alt−F2 drücken, gelangen Sie auf den zweiten Screen.

Sie können bis zu 12 Screen−Skripte für eine Workstation definieren. Den meisten Leuten reicht allerdings ein einziges.

Es gibt folgende Arten von Screen−Skripten:

startx

Dieses Skript startet den X−Server mit der Option "−query", um eine Anfrage an den Display−Manager zu senden, damit dieser dann eine Login−Dialogbox auf dem Bildschirm der Workstation darstellt.

shell

Dieses Skript startet eine Shell auf dem Terminal. Die ist zur Fehlersuche auf der Workstation gedacht. Da es sich um eine Session auf dem Thin Client und nicht auf dem Server handelt, ist sie nicht sehr nützlich, um Anwendungen zu starten.

telnet

Dieses Skript startet eine Telnet−Session mit dem Server. Hiermit erhalten Sie eine textbasierte Session auf dem Server.

In der Default−Einstellung verbindet sich telnet mit dem LTSP−Server. Wollen Sie einen anderen Server angeben, müssen Sie ihn zusammen mit dem Screen−Skript in der gleichen Zeile angeben. Dies sieht beispielsweise folgendermaßen aus:

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

Sie können auch Optionen, welche von telnet verstanden werden, angeben. Allerdings müssen Sie dann auch den Server, mit dem Sie sich verbinden wollen, mit angeben.

rdesktop

Dieses Skript startet ein rdesktop−Programm, mit dessen Hilfe Sie eine Verbindung zu einem Microsoft−Windows−Server herstellen können. Sie können jede von Ihnen gewünschte rdesktop−Option direkt nach der Bezeichnung des Screen−Skripts angeben. Wenn Sie z.B. eine Verbindung zu einem Server herstellen wollen, können Sie dies auf folgende Weise machen:

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

Im obigen Beispiel startet rdesktop im Vollbildmodus. Der Benutzer erhält einen Windows−Login−Dialog und muss sich nur einmal einloggen. Dies ist sehr nützlich, wenn man sofort ein Windows−Login ohne ein Linux−Login oder einen Fenstermanager bekommt. Der Benutzer weiß gar nicht, dass Linux ausgeführt wird.

Die Screen−Skripte liegen im Verzeichnis `/opt/ltsp/i386/etc/screen.d`. Sie können auch eigene Screen−Skripte erstellen und diese dann in jenem Verzeichnis ablegen. Wenn Sie ein eigenes Screen−Skript erstellen, ist es sehr ratsam, sich dabei an den bestehenden zu orientieren.

Kapitel 7. Fehlersuche

Für den Fall, dass die Workstation nicht bootet oder sonst etwas falsch läuft, sind hier einige Tipps zur Fehlersuche und −beseitigung.

Stellen Sie fest, bis wohin der Bootvorgang abläuft.

7.1. Fehlersuche beim Starten von Diskette

Einige Möglichkeiten, wie es beim Start von Diskette aussehen könnte:

```
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>
```

Wie das obige Beispiel zeigt, kann man den Startvorgang auf dem Monitor beobachten. Sieht man nichts, dann kann dies an einer fehlerhaften Diskette liegen oder daran, dass das Image nicht richtig auf die Diskette geschrieben wurde.

Bei einer Meldung wie der folgenden kann man vermuten, dass das von Ihnen erstellte Etherboot−Image nicht zu Ihrer Netzwerkkarte paßt.

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

Wenn die Netzwerkkarte erkannt wird und die MAC−Adresse richtig angegeben wird, dann ist die Diskette wohl in Ordnung und der Fehler liegt woanders.

7.2. DHCP−Fehlerursache

Sobald die Netzwerkkarte initialisiert wurde, schickt sie eine DHCP−Anfrage per Broadcast ins lokale Netz, um einen DHCP−Server zu finden.

Wenn die Workstation eine gültige Antwort erhält, dann konfiguriert sie die Netzwerkkarte. Wenn die IP−Adresse auf dem Monitor angezeigt wird, hat alles korrekt funktioniert. Hier ist ein Beispiel, wie es in etwa aussehen sollte:

```
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
```

Wenn Sie eine Zeile wie die letzte sehen ('Me:' gefolgt von einer IP−Adresse), dann arbeitet DHCP korrekt und der Fehler muss woanders liegen. Als nächstes sollte also überprüft werden, ob TFTP läuft.

Umgekehrt ist etwas falsch, wenn die folgende Meldung erscheint und danach jede Menge <sleep>−Meldungen erscheinen. Normal sind ein bis zwei <sleep>−Meldungen bis der DHCP−Server antwortet.

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

Es kann schwierig sein, die Fehlerursache herauszufinden. Im folgenden Abschnitt finden Sie hierzu ein paar Hinweise:

7.2.1. Überprüfen der Verbindungen

Ist die Workstation physikalisch richtig an dasselbe Netz wie der Server angeschlossen?

Kontrollieren Sie nach dem Einschalten der Workstation, ob die 'link'−LEDs der Netzwerkkarte überall aufleuchten.

Bei einer Direktverbindung zwischen Server und Workstation muss ein Cross−Over−Kabel verwendet werden, bei Einsatz von Hub oder Switch ausschließlich normale Patchkabel.

7.2.2. Arbeitet dhcpd?

Stellen Sie fest, ob der **dhcpd**−Prozess auf dem Server läuft. Dazu gibt es mehrere Möglichkeiten:

dhcpd läuft normalerweise im Hintergrund und wartet auf dem UDP−Port 67 auf Anfragen. Führen Sie das Kommando **netstat** aus, um zu sehen, ob der Prozess vorhanden ist:

```
netstat −an | grep ":67 "
```

Es sollte etwa folgendes zu sehen sein:

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

Die vierte Spalte zeigt, getrennt durch einen Doppelpunkt, die IP−Adresse und den Port an. Überall Nullen zeigen an, dass auf allen Netzwerkkarten Anfragen entgegengenommen werden. Gibt es also etwa **eth0** und **eth1**, dann wartet **dhcpd** auf beiden Interfaces auf Anfragen.

Allein die Tatsache, dass auf Port 67 ein Prozess auf Anfragen wartet, heißt noch nicht, dass dies **dhcpd** ist, es könnte auch **bootpd** sein. Dies ist allerdings unwahrscheinlich, da **bootpd** in den meisten Distributionen nicht mehr enthalten ist.

Um sicherzugehen, dass tatsächlich **dhcpd** läuft, führen Sie das Kommando **ps** aus.

```
ps aux | grep dhcpd
```

Dann sollte es etwa so aussehen:

```
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
```

Die erste Zeile gibt an: **dhcpd** läuft. Die zweite Zeile zeigt lediglich unser **grep**−Kommando.

Wenn Sie keine solche Zeile sehen, dann überprüfen Sie, ob das automatische Starten des DHCP−Servers im Runlevel 5 eingestellt ist. Auf Redhat−basierenden Systemen kann man dazu das Kommando **ntsysv** ausführen, in der erscheinenden Liste nach unten scrollen und sich vergewissern, dass **dhcpd** angekreuzt ist.

Sie können auch versuchen, den Prozess erst einmal von Hand zu starten. Das Kommando bei Fedora (Redhat) ist:

```
service dhcpd start
```

Achten Sie auf eventuelle Fehlermeldungen.

7.2.3. Blockieren ipchains oder iptables die Anfrage?

7.2.3.1. Checken von ipchains

Führen Sie das folgende Kommando aus und beobachten Sie die Ausgabe:

```
ipchains −L −v
```

Wenn Sie in etwa eine Ausgabe wie die folgende sehen, dann liegt es nicht an ipchains:

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

7.2.3.2. Checken von iptables

Führen Sie das folgende Kommando aus und beobachten Sie die Ausgabe:

```
iptables −L −v
```

Wenn Sie in etwa eine Ausgabe wie die folgende sehen, dann liegt es nicht an itables:

```
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
```

7.2.4. Sendet die Workstation die Anfrage?

Beobachten Sie auf dem Server die Datei `/var/log/messages`, während die Workstation bootet. Das geht mit dem folgenden Kommando:

```
tail −f /var/log/messages
```

Damit blättert der Dateiinhalt vor sich hin, jeder neue Eintrag wird sofort angezeigt.

```
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
```

Eine solche wie oben Meldung besagt, das **dhcpd** läuft, aber nichts über die anfragende Maschine weiß.

7.3. Fehlersuche bei TFTP

Etherboot benutzt TFTP, um den Linux−Kernel vom Server zu laden. Es handelt sich zwar um ein ziemlich einfaches Protokoll, aber es macht manchmal Probleme.

Wenn Sie auf der Workstation beim Startprozess in etwa folgendes sehen:

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

und sich der Bildschirm dabei schnell mit Punkten füllt, zeigt dies an, dass der Kernel geladen wird. TFTP sollte damit richtig laufen.

Füllt sich der Bildschirm nicht mit Punkten, weist dies auf ein Problem hin. Möglicherweise läuft das falsch, was im nächsten Abschnitt beschrieben wird.

7.3.1. tftpd läuft nicht

Wenn **tftpd** nicht so eingerichtet ist, dass er automatisch startet, dann kann er sicherlich keine Anfragen der Workstation entgegen nehmen und diese beantworten. Sie könnten das Programm **netstat** verwenden, um zu sehen, ob **tftpd** überhaupt ausgeführt wird. Dazu müssen Sie folgendes eingeben und erhalten dann in etwa diese Ausgabe:

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

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

Sehen Sie keine Ausgabe, nachdem Sie das Kommando ausgeführt haben, ist es wahrscheinlich, dass tftpd nicht läuft.

Es gibt für gewöhnlich zwei Methoden, um tftpd zu starten. Diese sind: **inetd** und das neuere **xinetd**.

inetd benutzt eine Konfigurationsdatei, die `/etc/inetd.conf` genannt wird. Stellen Sie sicher, dass die zum Start von tftpd erforderliche Zeile nicht auskommentiert ist. In etwa sollte diese folgendermaßen aussehen:

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

xinetd verwendet ein Verzeichnis mit darin enthaltenen individuellen Konfigurationsdateien. Jeder Dienst besitzt seine eigene Datei. Nutzt Ihr Server xinetd, heißt die Konfigurationsdatei für tftpd `/etc/xinetd.d/tftp`. Unten finden Sie ein Beispiel dafür, wie diese Datei aussehen kann:

```
service tftp
{
  disable          = no
  socket_type      = dgram
  protocol         = udp
  wait             = yes
  user             = root
  server           = /usr/sbin/in.tftpd
  server_args      = −s /tftpboot
}
```

Stellen Sie sicher, dass in der **disable**−Zeile nicht **yes** eingetragen ist.

7.3.2. Der Kernel befindet sich nicht an der von tftpd erwarteten Position

Der Kernel muss an einer Stelle liegen, auf die der tftpd−Daemon zugreifen kann. Wenn zum Starten von **tftpd** die '−s'−Option benutzt wird, dann müssen alle angegebenen Dateinamen relativ zu `/tftpboot` sein. Beispielsweise muss der Kernel sich im Verzeichnis `/tftpboot/lts/vmlinuz−2.4.24−ltsp−4` befinden, wenn der **filename**−Eintrag in der Datei `/etc/dhcpd.conf` auf `/lts/vmlinuz−2.4.24−ltsp−4` lautet.

7.4. Fehlersuche root−Dateisystem per NFS

Es gibt mehrere Dinge, die dem Mounten des root−Dateisystems im Wege stehen können:

7.4.1. Der init−Prozess fehlt

Bei der folgenden Fehlermeldung:

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

ist es sehr wahrscheinlich, dass entweder versucht wird, ein falsches root−Dateisystem zu mounten oder dass das Verzeichnis `/opt/ltsp/i386` leer ist.

7.4.2. Der Server meldet 'error −13'

Die Fehlermeldung:

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

zeigt an, dass entweder das Verzeichnis `/opt/ltsp/i386` nicht in der Datei `/etc/exports` enthalten ist oder dort fehlerhaft eingetragen wurde.

Sehen Sie in der Datei `/var/log/messages` nach irgendwelchen Hinweisen. Ein Eintrag wie dieser:

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

bestätigt z.B. den Verdacht, dass der Eintrag in `/etc/exports` fehlt.

7.4.3. NFS−Daemon−Probleme (portmap, nfsd & mountd)

Es kann schwierig sein, NFS−Fehler aufzuspüren. Wenn man versteht, was konfiguriert werden muss und welche Diagnosetools zur Verfügung stehen, wird es wohl etwas einfacher.

Auf dem Server müssen drei Prozesse laufen, damit NFS richtig funktioniert. Dies sind: **portmap**, **nfsd** und **mountd**.

7.4.3.1. Der Portmapper (portmap)

Bei folgenden Meldungen:

```
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
```

kann man vermuten, dass der Portmapper **portmap** nicht läuft. Feststellen lässt sich dies durch das **ps**−Kommando:

```
ps −e | grep portmap
```

Bei laufendem Portmapper sieht man in etwa dies:

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

Eine weitere Testmöglichkeit bietet das **netstat**−Kommando. Der Portmapper benutzt die TCP− und UDP−Ports 111. Führt man folgendes aus:

```
netstat −an | grep ":111 "
```

dann sollte dies zu sehen sein:

```
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:*
```

Sieht man nichts, dann läuft der Portmapper auch nicht. Sie starten den Portmapper per Hand auf folgende Weise:

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

Falls so der Portmapper gestartet werden kann, dann sollten Sie nun einstellen, dass er beim Hochfahren des Servers automatisch gestartet wird. Unter Redhat (Fedora) benutzen Sie dazu das Kommando **ntsysv**.

7.4.3.2. Die Prozesse NFS und MOUNT (nfsd & mountd)

Für NFS müssen zwei Daemon−Prozesse laufen: **nfsd** und **mountd**. Beide werden durch das Skript `/etc/rc.d/init.d/nfs` gestartet.

Mit dem Kommando **ps** können Sie feststellen, ob beide Prozesse laufen.

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

Falls einer oder beide Daemons nicht laufen, dann müssen diese nun gestartet werden.

Eigentlich sollte hier das **restart**−Argument nutzbar sein, aber auf diese Weise wird kein Neustart von **nfsd** erfolgen (Bug?). Deshalb sollten Sie folgende Kommandosequenz aufrufen:

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

Vielleicht gibt es Fehlermeldungen beim **stop**−Kommando, aber dies ist normal. Das **start**−Kommando sollte allerdings **OK** als Status zurückgeben.

Wenn die Prozesse laufen, NFS aber immer noch nicht funktioniert, dann kann mit dem Kommando **rpcinfo** feststellen, ob sie sich auch beim Portmapper angemeldet haben.

```
rpcinfo −p localhost
```

Sie sollten in etwa folgendes sehen:

```
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
```

Hier wird angezeigt, dass **nfs** (nfsd) und **mountd** laufen und sich beim Portmapper registriert haben.

7.5. Fehlersuche beim X−Server

Teufel auch! Die schwierigste Einzelaufgabe ist wahrscheinlich das richtige Konfigurieren des X−Servers einer LTSP−Workstation. Bei halbwegs neuer, von Xorg X−Servern unterstützter Graphikkarte und einem Monitor, der viele Frequenz− und Auflösungseinstellungen beherrscht, ist es kein Problem. Falls es nicht klappt, liegt es meist am falschen X−Server für die Karte.

Es lässt sich einfach feststellen, ob es der falsche X−Server ist: Entweder bricht der Start des Servers mit Fehlermeldungen ab oder die Anzeige sieht verzerrt oder irgendwie merkwürdig aus.

Ist die Workstation zum Start des X−Servers bereit, ruft sie das Skript `startx` auf, das den X−Server mit der **−query**−Option, die auf den Server verweist, auf dem ein Display−Manager wie **XDM**, **GDM** oder **KDM** läuft, startet.

Da der X−Server durch das Skript `startx` gestartet wird, welches selbst durch das **init**−Skript aufgerufen wird, versucht **init** bei nicht erfolgreichem Start des X−Servers, es wieder auszuführen. **init** versucht dies zehn mal, bevor es aufgibt ("respawning too fast"). Danach wird eine Fehlermeldung des X−Servers auf dem Monitor zu finden sein.

Es kann ziemlich irritierend sein, wenn man zehn mal sieht, wie der X−Server nicht gestartet werden kann. Ein einfacher Weg, um diese sich wiederholdenden Fehlversuche zu vermeiden, ist das Starten der Workstation im Runlevel 3, so dass kein automatischer Start des X−Servers erfolgt. Im Runlevel 3 hat man nach dem Start einen Shell−Prompt der **bash**. Der X−Server kann nun manuell mit dem folgenden Kommando gestartet werden:

```
sh  /tmp/start_ws
```

Statt nach zehn Versuchen gibt der X−Server nun nach einem auf und die Fehlermeldungen lassen sich bequem nachlesen.

7.6. Fehlersuche beim Display−Manager

Der Display−Manager ist ein auf dem Server laufender Prozess, der darauf wartet, von einem X−Server kontaktiert zu werden. Wenn dies passiert, dann schickt der Prozess ein Anmeldefenster auf den Bildschirm der Maschine, auf der der anfragende X−Server läuft. Benutzer können sich nun zur Nutzung von Programmen des Servers anmelden.

Die drei am häufigsten genutzten Display−Manager sind:

- XDM − Den gibt es schon seit ewigen Zeiten. Er gehört zum X−Window−System.
- GDM − Der 'Gnome Display−Manager'. Dieser gehört zum gnome−Paket.
- KDM − Der 'KDE Display Manager'. Und dieser ist Teil des KDE−Desktop−Systems

Neuere GNU/Linux−Distributionen haben alle drei im Angebot.

7.6.1. Grauer Bildschirm mit großem Cursor in Form eines X

Dies bedeutet, dass der X−Server läuft, aber keinen Kontakt zu einem Display−Manager herstellen konnte. Mögliche Gründe dafür sind:

1. Auf dem angesprochenen Rechner läuft kein Display−Manager

   Neuere Versionen von Redhat (7.0 und höher; auch Fedora), haben den Start des Display−Managers in den **init**−Prozess integriert. In der Datei `/etc/inittab` gibt es eine Zeile, die etwa so aussieht:

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

   Das **prefdm**−Skript legt fest, welcher Display−Manager gestartet werden soll.

   Welches der voreingestellte Display−Manager ist, hängt von den installierten Paketen ab. Ist Gnome installiert, dann ist GDM voreingestellt. Falls Gnome nicht installiert ist, prüft das prefdm−Skript, ob KDE installiert ist. Im zutreffenden Fall wird KDM als Display−Manager voreingestellt. Ist KDE ebenfalls nicht installiert, dann wird XDM zum voreingestellten Display−Manager.

   Mittels des **netstat**−Kommandos, lässt sich feststellen, welcher Display−Manager läuft. Auf dem Server gibt man folgenden Befehl ein:

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

   Es sollte zu sehen sein, dass ein Prozess auf dem xdmcp−Port (177) auf Anfragen wartet.

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

   Man sieht hier, dass **gdm** mit der PID 1493 läuft und auf dem xdmcp−Port bereit steht.

   In der Datei `lts.conf` lässt sich die IP−Adresse des Servers angeben, dessen Display−Manager um ein graphisches Login angefragt werden soll. Der Eintrag ist optional. Falls er vorhanden ist, sollte er so aussehen:

   ```
   XDM_SERVER  =  192.168.0.254
   ```

   Die IP−Adresse kann in Ihrem Fall natürlich anders sein.

Wenn der Eintrag 'XDM_SERVER' nicht vorhanden ist, dann wird der Eintrag 'SERVER' benutzt. Ist auch dieser nicht vorhanden, wird 192.168.0.254 genommen.

Was auch immer angegeben wird, es muss sich um die richtige IP−Adresse des Rechners handeln, auf dem der Display−Manager läuft.

2. Möglicherweise ist der Display−Manager so konfiguriert, dass Anfragen anderer Rechner aus Sicherheitsgründen abgewiesen werden.

   Wenn Sie sicher sind, dass der Display−Manager läuft, dann prüfen Sie nun, ob der Display−Manager vielleicht so wie gerade erwähnt konfiguriert wurde (XDMCP−Anfragen von entfernten Rechnern werden abgewiesen). Sie müssen die Konfigurationsdateien des laufenden Display−Managers überprüfen, um festzustellen, dass er korrekt konfiguriert wurde.

   - **XDM**

     Bei Redhat (Fedora) ist voreingestellt, dass Anfragen von anderen Rechnern abgelehnt werden. Das weit oben erwähnte Skript **ltspcfg** sollte eigentlich die notwendige Änderung an den Konfigurationsdateien vorgenommen haben. Falls dies nicht funktioniert hat, sollten Sie sich einmal die Datei `/etc/X11/xdm/xdm−config` ansehen. Suchen Sie nach einem Eintrag, der etwa so aussieht:

     ```
     DisplayManager.requestPort:     0
     ```

     Diese Zeile muss unbedingt auskommentiert werden, damit XDM Anfragen anderer Rechner über Port 177 entgegen nehmen kann.

     Damit XDM Anfragen anderer Rechner bedienen kann, ist noch der Inhalt einer weiteren Datei von Bedeutung: `/etc/X11/xdm/Xaccess`. In dieser Datei muss es unbedingt eine Zeile geben, die mit dem Zeichen '*' beginnt. Normalerweise gibt es diese Zeile, bei Redhat (Fedora) ist sie allerdings auskommentiert. Bei Problemen sehen Sie auch in dieser Datei nach. Eigentlich sollte dort solch eine Zeile zu sehen sein:

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

   - **KDM**

     Neuere Versionen von KDM benutzen eine Datei namens **kdmrc**. Leider legen die verschiedenen Linux−Distributionen diese Konfigurationsdatei an unterschiedlichen Positionen ab; bei Fedora Core 4 z.B. unter `/etc/X11/xdm/kdmrc`. Am besten benutzen Sie bei anderen Distributionen das Kommando **locate kdmrc**, um die Datei zu lokalisieren.

     Die Zugangskontrolle wird in dieser Datei in der Sektion **[Xdmcp]** festgelegt. Kontrollieren Sie, ob **Enable** auf **true** gesetzt ist.

   - **GDM**

     GDM benutzt andere Konfigurationsdateien. Diese befinden sich im Verzeichnis `/etc/X11/gdm`.

     Kontrollieren Sie in der Hauptdatei `gdm.conf` den Abschnitt **[xdmcp]**. Dort sollte ein Eintrag 'Enable' zu sehen sein. Dieser muss − je nach GDM−Version − auf '1' oder 'true'

gesetzt sein, wie in diesem Beispiel:

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

Beachten Sie die Zeile mit 'Enable=true'. Ältere Versionen von GDM benutzen '0' und '1', neuere 'false' und 'true' für die XDMCP−Zugangskontrolle.

3. Wenn feststeht, dass der Display−Manager läuft und auch so konfiguriert ist, dass Anfragen entfernter Rechner akzeptiert werden, dann kann die Fehlerursache noch folgende sein: Die Zuordnung von IP−Adressen zu Rechnernamen stimmt nicht. Der Name der Workstation muss entweder in der Datei `/etc/hosts` enthalten sein oder einen gültigen Eintrag in den Nameserver−Tabellen besitzen.

Kapitel 8. Kernel

Es müssen einige Entscheidungen über einen Kernel, der auf der Workstation laufen soll, getroffen werden. Man kann einen der vorbereiteten Kernel benutzen, die per Download verfügbar sind, oder man kann auch einen eigenen Kernel konfigurieren und kompilieren. Außerdem hat man die Wahl, ob beim Booten nur Textmeldungen oder ein Bild mit Fortschrittsbalken angezeigt werden soll. Letztere Variante wird durch den **Linux Progress Patch (LPP)** ermöglicht.

8.1. Von LTSP gelieferte Standard−Kernel

Das Kernel−Paket von LTSP enthält zwei Kernel, einen mit Linux Progress Patch und einen ohne.

Beide Kernel enthalten bereits den Patch, der Swap über NFS ermöglicht.

8.2. Einen Kernel selbst kompilieren

Es gibt zwei Wege, einen Kernel für LTSP zu konfigurieren: Die übliche Methode ist die Verwendung einer 'Initial Ram Disk' abgekürzt: **initrd**. Das initrd−Image ist ein kleines Dateisystem, welches an die Kerneldatei angehängt wird. Das initrd−Dateisystem wird in den Arbeitsspeicher geladen und wenn ein Kernel gebootet hat, mountet er diese RAM−Disk als sein root−Dateisystem. initrd bietet einige Vorteile: Wir können die Netzwerkkarten−Treiber als Modul kompilieren und beim Booten den passenden Treiber laden. Dies ermöglicht es, einen einzigen Kernel zu haben, der prinzipiell alle Netzwerkkarten unterstützt. Der andere Vorteil besteht darin, dass wir den DHCP−Client als "user−space"−Programm laufen lassen können, statt im "Kernel−space". Das DHCP−Client−Programm im "user−space" laufen zu lassen, ermöglicht eine bessere Kontrolle durch DHCP−Optionen, die vom Client angefordert und vom Server gesendet werden. Außerdem macht es den Kernel etwas kleiner. Der andere Weg, den Kernel zu konfigurieren, ist ohne initrd. Ein Kernel ohne initrd erfordert, dass der Netzwerkkarte−Treiber fest in den Kernel einkompiliert wird. Außerdem werden die Kernel−Config−Optionen "IP−Autoconfig" und "Root filesystem on NFS" benötigt. Ein Kernel ohne initrd ist etwas kleiner und bootet ein wenig schneller. Nachdem die Workstation gebootet ist, gibt es im laufenden Betrieb aber praktisch keinen Unterschied mehr.

Der Standard−Kernel des LTSP enthält eine initrd, welche die Erkennung der Netzwerkkarte übernimmt und eine "user−space" DHCP−Anfrage sendet. Es war ein wichtiges Ziel, dieses initrd−Image so klein wie möglich zu machen. Daher haben wir uClinux als Ersatz für die libc−Bibliothek und "busybox" für die Tools, welche wir während des Bootvorgangs benötigen, verwendet.

Wenn Sie eigene Kernel kompilieren wollen, sollten Sie das Paket ltsp_initrd_kit herunterladen. Es beinhaltet das root−Dateisystem und ein Skript, um ein Image zu erzeugen.

8.2.1. Woher bekommt man die Kernel−Quellen?

Wenn man sich einen eigenen Kernel baut, sollte man mit frischen, unveränderten Kernel−Quellen von **ftp.kernel.org** beginnen. Der Grund dafür ist, dass die Distributionen, wie z.B. Fedora, viele Patches zum Kernel hinzufügen, sodass deren Kernel−Quellen sich vom offiziellen Kernel unterscheiden.

Laden Sie das gewünschte Kernel−Paket herunter und speichern Sie es unter `/usr/src`. Die Kernel findet man unter `/pub/linux/kernel` auf ftp.kernel.org. Wir benötigen einen aktuellen 2.4.x Kernel, damit das **devfs**−Dateisystem unterstützt wird.

Wenn Sie NFS−Swapping oder dem Linux Progress Patch (LPP) benötigen, müssen Sie sicherstellen, dass die Patches zur Kernelversion passen. Zum Zeitpunkt der Dokumentationserstllung war 2.4.9 die neueste Kernel−Version, welche diese Features unterstützte.

Für unser Beispiel werden wir den Kernel 2.4.9 verwenden. Der Download−Pfad lautet:
`ftp://ftp.kernel.org/pub/linux/Kernel/v2.4/linux−2.4.9.tar.bz2`

Entpacken Sie die Kernel−Quellen in dem Verzeichnis `/usr/src`. Vorsicht: nach dem Auspacken befinden sich die Quellen in einem Verzeichnis mit dem Namen `linux`. Eventuell existiert bereits ein gleichnamiges Verzeichnis mit anderem Inhalt, und wir wollen das ja nicht vermischen. Deshalb sollten Sie prüfen, ob es schon ein Verzeichnis `linux` oder einen entsprechenden symbolischen Link gibt und dieses gegebenenfalls umbenennen oder verschieben, bevor Sie die neuen Quellen auspacken.

Das Quellpaket wurde mit **bzip2** komprimiert. Deshalb müssen wir es erst dekomprimieren bevor wir es mit **tar** auspacken:

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

Nach dem Auspacken haben Sie ein Verzeichnis mit dem Namen `linux`, welches den gesamten Sourcebaum enthält. Zu diesem Zeitpunkt geben wir dem Verzeichnis üblicherweise einen aussagekräftigen Namen.

```
mv linux linux−2.4.9
```

Nachdem das Verzeichnis umbenannt wurde, wechseln Sie hinein:

```
cd linux−2.4.9
```

Üblicherweise ändere ich das `Makefile`, bevor ich mit der Kernel−Konfiguration beginne. Ziemlich weit oben gibt es eine Variable mit dem Namen **EXTRAVERSION**. Diese stelle ich auf '−ltsp−1', sodass die Versionsnummer des neuen Kernels '2.4.9−ltsp−1' sein wird. Dies macht später die Unterscheidung einfacher. Der Anfang des Makefiles sollte dann so aussehen:

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

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

8.2.2. Kernel Patches

Nach dem Entpacken des Kernels haben Sie möglicherweise diverse Patches, die Sie einfügen wollen. Zum Beispiel den NFS−Swap−Patch oder den Linux Progress Patch. Man muss die Kernel−Quellen patchen BEVOR man den Kernel konfiguriert.

8.2.2.1. NFS−Swap−Patch

Der NFS−Swap−Patch ermöglicht der Workstation das Benutzen einer Swap−Datei, welche sich auf einem NFS−Server befindet. Üblicherweise wird empfohlen, genug RAM einzubauen, damit die Workstation nicht swappen muss. Aber manchmal ist es schwierig, mehr RAM einzubauen, besonders bei älteren Computern. In solchen Fällen macht das NFS−Swapping aus einem unbrauchbaren Rechner einen brauchbaren.

Wenn das aktuelle Verzeichnis /usr/src/linux−2.4.9 ist und der Patch sich in /usr/src befindet, kann man mit dem folgenden Kommando den Patch testen:

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

So können Sie herausfinden, ob der Patch zum Kernel passt. Wenn dieser Probelauf ohne Fehler endet, dann können Sie den Patch ohne die Option **−−dry−run** in die Quellen einfügen.

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

8.2.2.2. Linux Progress Patch (LPP)

Der Linux Progress Patch (LPP) ermöglicht es, dass ein graphisches Logo während des Bootens angezeigt wird. Die üblichen Boot−Meldungen werden auf eine andere Konsole umgeleitet und spezielle Anweisungen in den Boot−Skripten sorgen dafür, dass ein Fortschrittsbalken den Bootvorgang visualisiert.

Genau wie beim NFS−Swap−Patch kann man den LPP mit

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

testen. Wenn der Test erfolgreich war, wird der Patch durchgeführt:

```
patch −p1 <../lpp−2.4.9
```

8.2.3. Konfiguration von Kernel−Optionen

Nun können Sie das Kernel Konfigurationsprogramm ihrer Wahl starten. Zur Wahl stehen:

- make xconfig

  Dies ist die X−Window−Version des Kernel−Konfigurationstools.

- make menuconfig

  Dies ist eine Curses−basierte Version.

- make config

  Das ist die einfache Zeile−für−Zeile Version des Tools.

8.2.3.1. Kernel mit initrd konfigurieren

Die Verwendung von initrd erfordert die folgenden Optionen:

- File systems −> /dev filesystem support

  "/dev file system support" muss angewählt werden. Es befindet sich in der Rubrik 'File systems' (Dateisysteme). 'Automatically mount at boot' darf NICHT eingeschaltet seit, denn das Mounten wird durch das /linuxrc Script erledigt.

- Block devices −> RAM disk support

  LTSP−Workstations benötigen RAM−Disk−Support. Diese Option finden Sie in der Rubrik 'Block devices'.

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

Auch diese Option muss aktiviert werden.

- Processor type and features −> Processor family

  Sie müssen sicherstellen, dass der Kernel auf der CPU der Workstation läuft. Dies kann man in der Rubrik 'Processor type and features' einstellen. SMP−Support sollten Sie abschalten, es sei denn, ihre Workstation verfügt tatsächlich über mehrere CPUs.

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

  Die Workstation wird ihr Dateisystem via NFS mounten. Also muss diese Option aktiviert werden.

Dies waren die benötigten Optionen. Sie können noch viele Optionen abschalten, um die Größe des Kernels zu reduzieren.

8.2.3.2. Kernel ohne initrd konfigurieren

Das Konfigurieren eines Kernel ohne initrd unterscheidet sich vom Kernel mit initrd in einigen Punkten:

- Block devices −> RAM disk support

  LTSP−Workstations benötigen RAM−Disk−Support.

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

  Dies wird abgeschaltet.

- Networking options −> IP:Kernel level autoconfiguration

  Diese Option muss eingeschaltet werden. Dadurch konfiguriert der Kernel beim Booten automatisch die Ethernet−Schnittstelle eth0 anhand von Kernel−Bootparametern.

  Es ist nicht nötig die DHCP−, BOOTP− oder RARP−Optionen zu aktivieren, da der Etherboot−Code bereits eine DHCP− oder BOOTP−Anfrage gemacht hat und die IP−Parameter über die Kommandozeile an den Kernel übergibt. Deswegen muss der Kernel selbst keine eigene Anfrage mehr starten.

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

  Wenn man auf initrd verzichtet, muss man einen bestimmten Netzwerkkarten−Treiber auswählen, der zur in der Workstation verwendeten Netzwerkkarte passt. Dieser Treiber MUSS fest einkompiliert werden, weil die Ethernet Schnittstelle benötigt wird, bevor das root−Dateisystem gemountet wird. Dies ist ein großer Unterschied zu einem Kernel mit initrd.

- File systems −> /dev filesystem support

  Seit LTSP Version 2.09pre2 wird **devfs** benötigt, egal ob mit oder ohne initrd.

- File systems −> Automatically mount at boot

Wenn KEINE initrd verwendet wird, dann muss das /dev−Dateisystem beim Booten vom Kernel gemountet werden. Also muss diese Option aktiviert werden.

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

  Die Workstation wird ihr root−Dateisystem per NFS mounten, also ist "NFS client support" nötig.

8.2.4. Den Kernel kompilieren

Um die Arbeit zu erleichtern, befindet sich eine Kopie der `.config`−Datei im ltsp_initrd_kit−Paket. Sie können diese ins Verzeichnis `/usr/src/linux−2.4.9` kopieren.

Nachdem Sie alle gewünschten Kernel−Optionen an− und abgewählt haben, können Sie den Kernel kompilieren:

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

Man kann diese Kommandos auch in einer Zeile zusammenfügen:

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

Das doppelte & bedeutet, dass das hintere Kommando erst gestartet wird, wenn das vordere erfolgreich ausgeführt wurde usw.

Nach dem Kompilieren des Kernels befindet sich der neue Kernel im Verzeichnis `/usr/src/linux−2.4.9/arch/i386/boot/bzImage`.

8.2.5. Den Kernel für Etherboot vorbereiten (Tagging)

Damit Etherboot mit dem Kernel umgehen kann, muss er präpariert werden. Das nennt man 'Kernel tagging'. Das Tagging fügt zusätzlichen Code zum Kernel hinzu, der ausgeführt wird, bevor die Kontrolle an den Kernel übergeben wird. Das Programm für das Kernel−Tagging heißt '**mknbi−linux**'.

Das Paket ltsp_initrd_kit enthält ein Shell−Skript mit dem Namen **buildk**, welches alle Kommandos enthält, die man benötigt, um den Kernel für das Booten über das Netzwerk vorzubereiten.

Kapitel 9. lts.conf−Einträge

Als wir LTSP entwickelt haben, wussten wir, dass wir mit unterschiedlicher Hardware−Ausstattung bei den Workstations rechnen mussten. Egal welche Kombination aus Prozessor, Netzwerk− und Grafikkarte wir heute verwenden, wir konnten fast sicher sein, dass diese Kombination in drei Monaten, wenn wir mehr Workstations hinzufügen wollen, nicht mehr erhältlich sein würde.

Also haben wir uns einen Weg ausgedacht, wie man die Konfiguration aller Workstations zentral speichert. Diese Konfigurationsdatei heißt `lts.conf` und befindet sich im Verzeichnis `/opt/ltsp/i386/etc`.

Das Format von lts.conf erlaubt Standardeinstellungen ('default') und davon abweichende Einstellungen für einzelne Workstations. Wenn Ihre Workstations alle identisch sind, dann genügt es, alle Einstellungen im Abschnitt '[Default]' einzutragen.

9.1. Eine ltsp.conf−Beispieldatei

Hier sehen Sie ein Beispiel dafür, wie diese Datei aussehen könnte:

```
[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. In ltsp.conf verfügbare Parameter

9.2.1. Allgemeine Parameter

Kommentare
: Kommentare beginnen mit dem '#' Zeichen und gelten bis ans Zeilenende.

LTSP_BASEDIR
: Damit gibt man das LTSP−root−Dateisystem an. Der Defaultwert ist `/opt/ltsp`.

SERVER

Dies ist der Server, der als XDM_SERVER, TELNET_HOST, XFS_SERVER und SYSLOG_HOST, genommen wird, falls er nicht für einzelne Dienste extra angegeben wird. Wenn Sie also einen Rechner haben, der als Server für alles fungiert, dann können Sie diesen hier angeben und die anderen Server−Parameter weglassen. Wenn dieser Wert nicht gesetzt wird, wird der Default−Wert **192.168.0.254** benutzt.

SYSLOG_HOST
Wenn Sie Log−Meldungen an eine andere Maschine als den Default−Server senden wollen, dann können Sie diese hier angeben. Wenn dieser Wert NICHT gesetzt wird, dann wird der **SERVER**−Parameter, wie oben beschrieben, verwendet.

NFS_SERVER
Hier können Sie die IP−Adresse des NFS−Servers angeben, der zum Mounten des `/home`−Verzeichnisses benutzt werden soll. Wenn dieser Wert nicht gesetzt wird, dann wird der **SERVER**−Parameter verwendet.

USE_NFS_SWAP
Setzen Sie diese Option auf **Y**, wenn Sie NFS−Swap verwenden wollen. Der Defaultwert ist **N**.

SWAPFILE_SIZE
Hier können Sie die Größe der Swap−Datei angeben. Der Defaultwert ist **64m**.

SWAP_SERVER
Die Swap−Datei kann auf einem beliebigen NFS−Server im Netzwerk liegen. Hier können Sie dessen IP−Adresse angeben. Der Defaultwert entspricht dem Wert, der mit **NFS_SERVER** angegeben wird.

NFS_SWAPDIR
Der Pfad, der vom NFS−Server für Swap−Dateien freigegeben wurde. Der Defaultwert ist `/var/opt/ltsp/swapfiles`. Achten Sie darauf, dass dieses Verzeichnis beim NFS−Server in `/etc/exports` eingetragen ist.

TELNET_HOST
Wenn die Workstation als Text−Terminal genutzt wird, dann kann man hier angeben, zu welchem Server eine Telnet−Verbindung aufgebaut werden soll. Wenn dieser Wert nicht gesetzt ist, wird der Wert von **SERVER** genutzt.

DNS_SERVER
Anhand dieser Angabe wird die Datei `resolv.conf` der Workstation generiert.

SEARCH_DOMAIN
Auch dieser Wert landet in der Datei `resolv.conf` der Workstation.

SCREEN_01 *bis* ***SCREEN_12***
Bis zu 12 Screen−Skripte können durch diese Einträge geladen werden. Hierdurch erhalten Sie bis zu 12 Sessions auf einer Workstation. Jede kann durch das Drücken der Tasten Strg−Alt−F1 bis Strg−Alt−F12 erreicht werden.

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

Zurzeit können folgende Werte verwendet werden:

- startx
- telnet
- rdesktop
- shell

Schauen Sie in das Verzeichnis `/opt/ltsp/i386/etc/screen.d`, um mehr Screen−Skripte zu finden oder Sie legen Ihre eigenen hier ab.

MODULE_01** bis **MODULE_10

Bis zu zehn Kernel−Module können durch diese Einträge geladen werden. Die Einträge entsprechen dem, was man hinter insmod angeben würde. Beispielsweise:

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

Wenn man hier einen absoluten Pfad angibt, wird **insmod** benutzt, um das Modul zu laden. Ansonsten wird **modprobe** verwendet.

RAMDISK_SIZE

Wenn eine Workstation bootet, erzeugt sie eine RAM−Disk und mountet sie nach `/tmp`. Die Größe dieser RAM−Disk kann man mit diesem Parameter beeinflussen. Angaben werden in kByte (1024Bytes)−Einheiten gemacht. Um eine 1MB RAM−Disk zu erzeugen, schreiben Sie **RAMDISK_SIZE = 1024**.

Wenn Sie die Größe der RAM−Disk hier ändern, müssen Sie sie auch im Kernel verändern. Entweder fest einkompiliert oder beim Einsatz von Etherboot oder Netboot kann man die Größe angeben, wenn man den Kernel mit mknbi−linux 'taggt'.

Der Defaultwert ist 1024 (= 1 MB).

RCFILE_01** bis **RCFILE_10

Das rc.local−Skript kann zusätzliche Startskripte ausführen. Legen Sie diese Skripte einfach ins `/etc/rc.d`−Verzeichnis und geben Sie hier den Namen ihres Skripts an.

SOUND

Wenn das LTSP−Sound−Paket installiert wurde, dann muss dieser Wert auf **Y** stehen, damit das Skript **rc.sound** gestartet wird. rc.sound konfiguriert die Soundkarte und den Sound−Deamon (Sound−Server). Der Defaultwert ist **N**.

9.2.2. X−Window−Einstellungen

XDM_SERVER

Wenn die XDM−Anfrage an einen anderen als den Standard−Server gesendet werden soll, dann kann man diesen hier angeben. Lässt man diese Parameter weg, dann wird der Wert von 'SERVER' genommen.

XSERVER

Hier kann man den X Server angeben, der auf der Workstation verwendet werden soll. Bei PCI− und AGP−Karten sollte dieser Parameter nicht nötig sein. Das rc.local−Skript ist üblicherweise in der Lage, die Grafikkarte automatisch zu erkennen. Um dies zu verdeutlichen, kann man den Wert auch auf **auto** stellen.

Für ISA−Grafikkarten oder um einen speziellen Treiber zu wählen, kann man hier den Namen des Treiber−Moduls oder den X−Server angeben.

Beginnt der Wert mit **XF86_**, wird XFree86 3.3.6 verwendet, ansonsten X.org 6.7.0. Der Default−Wert ist **auto**

X_MODE_0 *bis* ***X_MODE_2***

Bis zu drei Modelines oder Bildschirm−Auflösungen kann man hier angeben. Für jeden Mode kann man entweder eine Auflösung oder eine komplette Modeline angeben.

```
X_MODE_0 = 800x600

   or

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

Wenn keine X_MODE_x Angaben gemacht werden, dann werden Standard−Modelines und die Auflösungen 1024x768, 800x600 und 640x480 verwendet.

Wenn eine oder mehrere X_MODE_x Angaben gemacht werden, dann wird keine der Standard−Modelines mehr verwendet.

X_MOUSE_PROTOCOL

Jeder Wert, der von X.org als Pointer Protocol bekannt ist, kann hier verwendet werden. Typische Werte sind z.B. "Microsoft" und "PS/2". Der Defaultwert ist **"PS/2"**.

X_MOUSE_DEVICE

Dies ist die Gerätedatei für den Maus−Anschluss. Bei seriellen Mäusen z.B. **/dev/ttyS0** oder **/dev/ttyS1**. Beim PS/2 Anschluss ist es **/dev/psaux**. Der Defaultwert ist **/dev/psaux**.

X_MOUSE_RESOLUTION

Dies ist der 'Resolution'− (Auflösung)−Wert für die Maus in der **xorg.conf**− bzw. **XF86Config**−Datei. Ein typischer Wert für serielle Mäuse ist **50** und für eine PS/2 Maus **400**. Der Defaultwert ist **400**.

X_BUTTONS

Hier kann man die Anzahl der Mausknöpfe angeben. Üblicherweise **2** oder **3**. Der Defaultwert ist **3**.

X_MOUSE_EMULATE3BTN

Durch den Wert **Y** kann man die mittlere Maustaste bei 2−Tasten−Mäusen emulieren, indem man beide Tasten gleichzeitig drückt. Der Defaultwert ist **N**.

X_MOUSE_BAUD

Hier kann man die Baudrate für serielle Mäuse angeben. Der Defaultwert ist **1200**.

X_COLOR_DEPTH

Damit kann man die Farbtiefe angeben. Mögliche Werte sind **8** (= 256 Farben), **15**, **16** (= 65536 Farben), **24** und **32** (= 4,2 Milliarden Farben). Nicht alle X−Server bieten alle Farbtiefen. Der Defaultwert ist **16**.

USE_XFS

Bei den Fonts (Zeichensätzen) kann man entweder einen X−Font−Server verwenden oder die Zeichensätze per NFS lesen. Der Font−Server soll die Verwaltung von Fonts an einer zentralen Stelle erleichtern. Allerdings gab es Probleme bei mehr als 40 Workstations. Die zwei möglichen Werte lauten hier **Y** und **N**. Der Defaultwert ist **N**. Wenn Sie einen Fontserver verwenden wollen, dann können Sie mit **XFS_SERVER** die IP des Fontservers angeben.

XFS_SERVER
Wenn man einen X−Font−Server benutzt, dann kann man hier seine IP−Adresse angeben. Wenn diese Variable nicht gesetzt ist, wird der Wert von **SERVER** genommen.

X_HORZSYNC
Hier kann man den **HorizSync**−Parameter für X.org angeben. Der Defaultwert ist **"31−62"**.

X_VERTREFRESH
Hier kann man den **VertRefresh**−Parameter für X.org angeben. Der Defaultwert ist **"55−90"**.

XF86CONFIG_FILE
Wenn Sie eine eigene komplette Xorg.conf−Datei vorbereitet haben, dann können Sie diese im Verzeichnis `/opt/ltsp/i386/etc` ablegen. Geben Sie der Datei einen aussagekräftigen Namen, z.B. xorg.conf.ws004.

```
XF86CONFIG_FILE = xorg.conf.ws004
```

9.2.3. Touch−Screen−Parameter

USE_TOUCH
Wenn die Workstation über einen Touch Screen verfügt, können Sie dies hier durch **Y** aktivieren. Der Defaultwert ist **N**.

X_TOUCH_DEVICE
Ein TouchScreen arbeitet ähnlich wie eine Maus und wird üblicherweise an einem seriellen Port angeschlossen. Diesen Anschluss können Sie hier angeben, z.B. **/dev/ttyS0**. Für diesen Eintrag existiert kein Defaultwert.

X_TOUCH_MINX
Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **433**.

X_TOUCH_MAXX
Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **3588**.

X_TOUCH_MINY
Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **569**.

X_TOUCH_MAXY
Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **3526**.

X_TOUCH_UNDELAY
Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **10**.

X_TOUCH_RPTDELAY

Kalibrierungswert für einen EloTouch Touch−Screen. Defaultwert: **10**.

9.2.4. Parameter für lokale Anwendungen

LOCAL_APPS
Wenn Anwendungen lokal auf einer Workstation laufen sollen, stellen Sie diesen Wert auf **Y**. Diverse weitere Einstellungen müssen am Server vollzogen werden, um Workstation−lokale Anwendungen zu ermöglichen. Lesen Sie hierzu das Kapitel 'Lokale Anwendungen' im LTSP−Handbuch. Der Defaultwert ist **N**.

NIS_DOMAIN
Wenn Sie (Workstation−)lokale Anwendungen nutzen wollen, benötigen Sie einen NIS−Server in ihrem Netz. Hier können Sie den Namen der NIS−Domain angeben. Der Name muss mit dem Namen, der im NIS−Server eingestellt ist, übereinstimmen. Eine NIS−Domain ist nicht dasselbe wie eine Internet−Domain. Der Defaultwert ist **ltsp**.

NIS_SERVER
Geben Sie hier die IP−Adresse des NIS−Servers an, wenn die Workstation keinen Broadcast senden soll, um einen entsprechenden NIS−Server zufinden.

9.2.5. Tastatur−Parameter

Alle Dateien zur (internationalen) Tastatur−Unterstützung sind im LTSP−Verzeichnisbaum (`/opt/ltsp/i386`) vorhanden. Deshalb ist die Konfiguration der Tastatur nur noch eine Sache von Einträgen in der X.org−Konfiguration. Hier stehen diverse Parameter zur Verfügung.

Die Bezeichnungen und möglichen Werte der folgenden Parameter stammen aus der Dokumentation von X.org. Alle für X.org gültigen Werte sind auch hier gültig.

Wir würden in Zukunft gerne weitere Abschnitte hinzufügen, die auflisten, welche konkreten Einstellungen für die einzelnen internationalen Tastaturen benötigt werden. Wenn es ihnen gelingt, ihre länderspezifische Tastatur einzustellen, dann informieren Sie bitte das LTSP Entwickler−Team.

XkbTypes
Der Defaultwert ist das Wort '**default**'.

XkbCompat
Der Defaultwert ist das Wort '**default**'.

XkbSymbols
Der Defaultwert ist '**us(pc101)**'. Dieser Defaultwert ist auch für deutsche Tastaturen o.k., eventuell auf us(pc105) setzen.

XkbModel
Der Defaultwert ist '**pc101**'. Dieser Defaultwert ist auch für deutsche Tastaturen o.k., eventuell auf pc105 setzen.

XkbLayout
Der Defaultwert ist '**us**'. Der Wert für eine deutsche Tastatur lautet '**de**'.

XkbVariant
Der Defaultwert ist die leere Zeichenkette '**nodeadkeys**'. Der Wert könnte für eine deutsche Tastatur auf '**nodeadkeys**' gesetzt werden.

9.2.6. Parameter für die Druckerkonfiguration

Maximal drei Drucker können an einer Workstation angeschlossen werden. Eine Kombination aus seriellen und parallelen Druckern kann mit den folgenden Einträgen in der Datei **lts.conf** konfiguriert werden:

PRINTER_0_DEVICE
Das Device (Anschluss) des ersten Druckers. Folgende Namen wie **/dev/lp0**, **/dev/ttyS0** oder **/dev/ttyS1** sind erlaubt.

PRINTER_0_TYPE
Der Druckertyp. Gültige Werte sind '**P**' für Parallel−Port−Drucker, und '**S**' für serielle Drucker.

PRINTER_0_PORT
Die TCP/IP−Port−Nummer für den Druck−Server−Prozess auf der Workstation (HP JetDirect Protokoll). Defaultwert ist '**9100**'.

PRINTER_0_SPEED
Für serielle Drucker kann man hier die Baudrate einstellen. Der Defaultwert ist '**9600**'.

PRINTER_0_FLOWCTRL
Für serielle Drucker kann man hier die Art der Flusskontrolle angeben. Entweder '**S**' für Software−(XON/XOFF)−Flusskontrolle, oder '**H**' für Hardware−(CTS/RTS)−Flusskontrolle. Wenn nichts angegeben wird, gilt '**S**' als Defaultwert.

PRINTER_0_PARITY
Für serielle Drucker kann man hier die Parität angeben. Zur Auswahl stehen: '**E**'−Even(Gerade), '**O**'−Odd(Ungerade) oder '**N**'−None(keine Parität). Wenn nichts angegeben wird, gilt '**N**' als Defaultwert.

PRINTER_0_DATABITS
Für serielle Drucker kann man hier die Anzahl der Datenbits angeben. Zur Auswahl stehen: '**5**', '**6**', '**7**' und '**8**'. Wenn nichts angegeben wird, gilt '**8**' als Defaultwert.

PRINTER_1_DEVICE
Device (Anschluss) des zweiten Druckers

PRINTER_1_TYPE
Typ des zweiten Druckers

PRINTER_1_PORT
TCP/IP−Port des zweiten Druckers

PRINTER_1_SPEED
Baudrate des zweiten Druckers (seriell)

PRINTER_1_FLOWCTRL
Flusskontrolle des zweiten Druckers(seriell)

PRINTER_1_PARITY
Parität des zweiten Druckers(seriell)

PRINTER_1_DATABITS
Anzahl Datenbits des zweiten Druckers(seriell)

PRINTER_2_DEVICE
Device−Name des dritten Druckers

PRINTER_2_TYPE
Typ des dritten Druckers

PRINTER_2_PORT
TCP/IP−Port des dritten Druckers

PRINTER_2_SPEED
Baudrate des dritten Druckers (seriell)

PRINTER_2_FLOWCTRL
Flusskontrolle des dritten Druckers(seriell)

PRINTER_2_PARITY
Parität des dritten Druckers(seriell)

PRINTER_2_DATABITS
Anzahl Datenbits des dritten Druckers(seriell)

Kapitel 10. Lokale Anwendungen

- ***Hinweis der Übersetzer:*** Dieser Abschnitt trifft in der Gesamtheit nur für LTSP−Versionen vor 4.0 zu. Insbesondere wird ab Version 4.0 für das Starten lokaler Anwedungen statt **rsh** das mehr Sicherheit bietende **ssh**−Kommando verwendet. Die Einrichtung ist damit allerdings noch aufwendiger geworden. Sehen Sie die folgenden Ausführungen daher eher als Hintergrundinformation an; Sie können im *LTSP−Wiki* nachlesen, wenn Sie sich über das konkrete Vorgehen beim Ausführen lokaler Anwendungen informieren möchten.

In einer LTSP−Umgebung hat man die Wahl, ob die Programme lokal auf der Workstation oder remote auf dem Server laufen sollen.

Die Anwendungen auf dem Server laufen zu lassen, ist die einfachere Variante. Dabei läuft die Anwendung auf dem Prozessor und im Arbeitsspeicher des Servers, die Interaktion mit dem Benutzer, also Bildschirmausgabe und Tastatur− und Mauseingabe, erfolgt auf der Workstation.

Dies ist eine grundlegende Fähigkeit des X−Window−Systems. Die Workstation funktioniert wie ein X−Window−Terminal.

Wenn eine Anwendung auf der Workstation ausgeführt werden soll, dann benötigt die Workstation einige Informationen über den Benutzer:

- die Benutzer ID (UID)
- die primäre Benutzergruppe (GID)
- das Heimatverzeichenis des Benutzers

LTSP benutzt den Network Information Service − NIS (früher auch ***Yellow Pages genannt***), um den Workstations diese Informationen zur Verfügung zu stellen.

10.1. Vorteile von lokalen Anwendungen

Es gibt einige Vorteile bei der lokalen Ausführung von Anwendungen:

- Reduzierte Server−Last. In großen Netzen mit speicherintensiven Anwendungen, wie z.B. Mozilla, kann es die Performance verbessern, solange die Workstation über genügend CPU−Leistung und RAM verfügt, um die Anwendung(en) zu bewältigen.
- Sog. "runaway" Prozesse, also Prozesse, die aufgrund von Fehlern viel CPU−Leistung und RAM an sich reißen, wirken sich nicht störend auf andere Benutzer aus.
- Die Soundunterstützung ist viel einfacher zu konfigurieren, wenn die Anwendung, die den Sound erzeugt, lokal auf der Workstation läuft.

10.2. Dinge, die man beim Einsatz von lokalen Anwendungen beachten sollte

Lokale Anwendungen erfordern deutlich mehr Konfigurationsaufwand.

- Lokale Anwendungen erfordern mehr Leistung von der Workstation−Hardware. Man benötigt mehr RAM und eine schnellere CPU. Empfehlung: 64MB RAM oder mehr.

- NIS − um Anwendungen auf der Workstation auszuführen, muss man sich zuerst gegenüber der Workstation identifizieren. Sie muss wissen, wer Sie sind. Dazu wird eine Art von Passwort−Authentifizierung benötigt. Wir haben NIS gewählt, um eine Authentifizierung übers Netzwerk zu ermöglichen.
- Für lokale Anwendungen müssen zusätzliche Verzeichnisse vom Server per NFS exportiert werden.
- Anwendungen starten langsamer, denn sie müssen via NFS gelesen werden, was außerdem auch mehr Netzwerk−Traffic verursacht. Weil jede Kopie eines Programms auf seiner eigenen CPU läuft, verzichtet man auf den Vorteil von gemeinsamen (shared) Code−Segmenten im RAM. Gemeinsame Code−Segmente sorgen dafür, dass weitere Instanzen eines Programms deutlich schneller starten.

10.3. Server−Konfiguration für lokale Anwendungen

10.3.1. Einträge in lts.conf

In der `lts.conf`−Datei müssen einige Einträge gemacht werden:

LOCAL_APPS
: Dieser Wert muss auf **Y** stehen. Dadurch werden folgende Vorgänge beim Booten der Workstation ausgelöst:

  1. Das `/home`−Verzeichnis des Servers wird per NFS gemountet.
  2. Die Datei `/var/yp/nicknames` wird auf der Workstation angelegt.
  3. Der **portmapper** wird auf der Workstation gestartet.
  4. **xinetd** wird auf der Workstation gestartet.
  5. Die Datei `/etc/yp.conf` wird auf der Workstation erzeugt.
  6. Das **domainname**−Kommando wird mit dem Wert von **NIS_DOMAIN** aus lts.conf ausgeführt.
  7. Das Kommando **ypbind** wird ausgeführt.

NIS_DOMAIN
: Ein NIS−Client versucht entweder eine Verbindung mit einem bestimmten NIS−Server herzustellen, oder er sendet einen Broadcast ins Netz und wartet, dass sich ein Server meldet. Wenn Sie einen NIS−Server angeben wollen, dann tragen Sie seine IP−Adresse in NIS_SERVER ein.

10.3.2. Network Information Service − NIS

NIS ist ein Client/Server−Dienst. Auf dem Server läuft ein Daemon (Hintergrundprozess/Dienst), welcher Anfragen von den Workstations entgegen nimmt. Dieser Prozess heißt **ypserv**.

Auf der Workstation gibt es den Prozess **ypbind**. Wenn die Workstation Informationen über den User benötigt, z.B. zur Passwort−Überprüfung, oder den Namen des Heimatverzeichnisses, dann wird ypbind benutzt, um eine Verbindung mit ypserv auf dem Server herzustellen.

Wenn Sie bereits NIS in ihrem Netzwerk verwenden, ist es nicht nötig, den LTSP−Server als zusätzlichen NIS−Server zu betreiben. Tragen Sie einfach in lts.conf für NIS_DOMAINNAME und NIS_SERVER Werte ein, die zu Ihrem Netz passen.

Wenn Sie in ihrem Netz bisher noch kein NIS verwenden, müssen Sie **ypserv** auf dem Server konfigurieren und starten.

Umfassende Informationen über die Installation eines NIS−Servers finden Sie im ***The Linux NIS(YP)/NYS/NIS+ HOWTO*** des LDP (Linux Documentation Project). Siehe "Weitere Informationsquellen" am Ende dieses Handbuchs.

10.4. Konfiguration von Anwendungen

Damit eine Anwendung auf der Workstation laufen kann, müssen Sie alle Teile der Anwendung dahin kopieren, wo die Workstation Sie auch finden kann.

Bei älteren LTSP−Versionen (2.08 und früher) wurden viele Verzeichnisse des Servers per NFS exportiert und von der Workstation gemountet. Verzeichnisse wie z.B. `/bin`, `/usr/bin`, `/lib` und `/usr` wurden exportiert.

Das Problem bei dieser Vorgehensweise ist, dass es nur funktioniert, wenn Workstation und Server die gleiche Hardware−Architektur verwenden. Sogar der Unterschied zwischen einem Server mit Pentium II (i686) und einer Workstation mit Pentium I (i586) kann zu einem Problem werden, da der Server wahrscheinlich i686−Libraries (Programm−Bibliotheken) hat und keine i386−, i486− oder i586−Libraries.

Der sauberste Weg ist also, einen kompletten Verzeichnisbaum zu haben, der alle Programme und Libraries enthält, den die Workstation benötigt, unabhängig von den Programmen und Libraries des Servers.

Alle Teile, die eine (Workstation−lokale) Anwendung benötigt, müssen in diesem Verzeichnisbaum abgelegt werden. Eines der LTSP−Pakete, die auf der LTSP−Website zum Download zu Verfügung stehen, ist das Paket local_netscape. Es installiert viele Dateien ins `/opt/ltsp/i386/usr/local/netscape`−Verzeichnis. Darin enthalten sind Java−Klassen, Hilfe−Texte, Programmdateien und Skripte.

Netscape benötigt keine zusätzlichen System−Libraries, deswegen muss in `/opt/ltsp/i386/lib` nichts hinzugefügt werden. Viele andere Anwendungen benötigen jedoch zusätzliche Libraries.

Also: wie finden wir heraus, welche Libraries benötigt werden? Dazu lässt sich das **ldd**−Kommando gut verwenden.

Nehmen wir an, Sie wollen eine bestimmte Anwendung als Workstation−lokale Anwendung einrichten. Als Beispiel nehmen wir das Programm **gaim**. gaim ist ein AOL−Instant−Messenger−Client, er ermöglicht die Kommunikation mit anderen AIM−Benutzern im Internet.

Als erstes müssen Sie die gaim Binärdatei finden. Bei Fedora liegt sie im `/usr/bin`−Verzeichnis. Die Tools **type** und **which** sind für eine solche Suche hilfreich.

Wenn Sie die **gaim**−Binärdatei gefunden haben, können Sie mit **ldd** die benötigten Libraries herausfinden:

```
[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)
```

Das obere Listing zeigt alle dynamisch gelinkten Libraries, die von **gaim** benötigt werden.

Die meisten Programme, die shared Libraries verwenden, benötigen den dynamischen Loader **ld−linux** um jede der Libraries zu finden und zu laden. Manche Programme laden die Libraries jedoch manuell mit dem **dlopen()**−Systemaufruf. Bei solchen Anwendungen wird **ldd** die benötigten Libraries nicht anzeigen. Für solche Fälle kann man das **strace**−Kommando verwenden. strace wird alle **dlopen()**−Aufrufe mit den zugehörigen Libraries anzeigen.

Hat man die Liste der benötigten Libraries ermittelt, müssen diese an die richtigen Stellen in den `/opt/ltsp/i386`−Baum kopiert werden.

10.5. Lokale Anwendungen starten

X−Window−Programme starten üblicherweise dort, wo der Fenstermanager läuft. Wenn der Fenstermanager auf dem Server läuft und seine Ausgaben auf die Workstation schickt, dann wird jede gestartete Anwendung ebenfalls auf dem Server laufen und die Ausgaben zur Workstation schicken.

Der Trick bei lokalen Anwendungen besteht darin, dass der Server die Workstation anweist, ein Programm zu starten. Dies wird üblicherweise mit dem **rsh**−Kommando gemacht.

Hier ein Beispiel wie man **gaim** auf der Workstation startet:

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

Das obige Beispiel kann man in einem **xterm**−Fenster eingeben, oder in ein Shell−Skript schreiben, welches dann durch Klick auf das entsprechende Icon ausgelöst werden kann.

Der lokale Start von Netscape läuft ähnlich ab, allerdings muss vor dem Start eine Umgebungs−Variable gesetzt werden.

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

Kapitel 11. Konfigurations−Beispiele

Fast alle Eigenschaften der Workstation können durch einen Eintrag in der Datei `lts.conf` festgelegt werden. Diese Datei liegt gewöhnlich im Verzeichnis `/opt/ltsp/i386/etc`.

11.1. Serielle Maus

Beispieleintrag in `lts.conf` für eine serielle Standard−Maus mit zwei Tasten:

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

11.2. PS/2 Wheel Maus

Beispieleintrag in `lts.conf` für eine 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. USB−Drucker an ThinkNic

Der ThinkNIC−Client hat einen USB−Anschluss, der für einen Drucker benutzt werden kann. Beispieleintragungen für solch einen Drucker in der `lts.conf`−Datei:

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

11.4. Die Workstation zum Laden eines XFree86 3.3.6. X−Servers zwinigen

Per Voreinstellung wird X.org 6.7.0 für die Workstation eingerichtet. Für ältere Grafikkarten, die von dieser X−Server−Version nicht mehr unterstützt werden, muss zunächst das richtige X−Server−Paket der Version 3.3.6 heruntergeladen und installiert werden. Dann muss man noch die richtigen Einträge in der Datei `lts.conf` vornehmen. Beispieleinträge für den Start des **SVGA**−Xservers (dieser unterstützt sehr viele alte Karten):

```
XSERVER             = XF86_SVGA
```

Kapitel 12. Weitere Informationsquellen

12.1. Im Internet

1. Die LTSP−Homepage

   www.LTSP.org

2. Diskless−Nodes HOW−TO−Dokument für Linux

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

3. Etherboot−Homepage

   etherboot.sourceforge.net

4. Die Rom−O−Matic−Webseite

   www.Rom−O−Matic.net

5. X.org−Maus−Unterstützung

   www.xfree86.org/current/mouse.html

6. XFree86−Video−Timings−HOWTO

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

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

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

12.2. Gedruckte Publikationen

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)