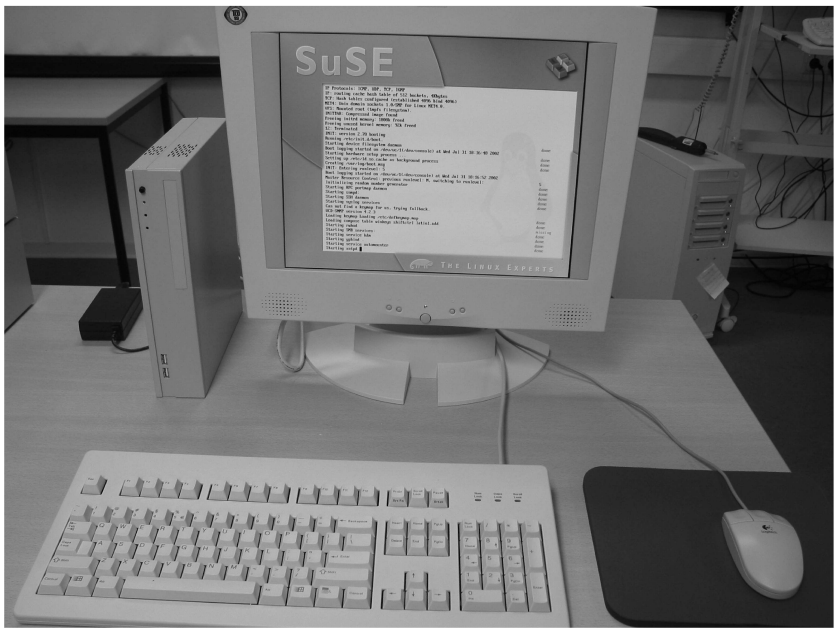
TUX in der Arztpraxis:

Linux Diskless Clients

Aufbau und Betrieb von Diskless X-Stations

Erfahrungsbericht, Grundlagen und Realisation

Karl-Heinz Heggen und Dirk von Suchodoletz



MULTI-DATA und
Rechenzentrum Universität Freiburg (RUF)

11. Mai 2003

this page was left blank intentionally

Inhaltsverzeichnis

1 Einleitung **4**
1.1 Ein Erfahrungsbericht 4

2 Grundsätzliche Idee der Diskless X-Station **6**

3 Einsatzgebiete von DXS **7**

4 Designüberlegungen **8**

5 Protokolle und Technologien **8**

6 Die Client-Boot-Software **10**
6.1 Vorüberlegungen 10
6.2 Etherboot 10
6.3 PXE 12
6.4 Syslinux und PXE 13

7 DHCP - Das zentrale Konfigurationswerkzeug **13**
7.1 Überblick 13
7.2 Server 13
7.3 DHCP-Standardoptionen 14
7.4 Benutzerdefinierte Optionen 15
7.5 Die Verwendung von Vendor-Code-Identifiern 15
7.6 DHCP-Client 16

8 Dateisystem und Softwareverwaltung **17**
8.1 Aufteilung und Dateisysteme 17
8.2 Aufsetzen des Dateisystems 18
8.3 Das INITTAR 19

9 Software-Anpassungen **19**
9.1 Kernel 19
9.2 Programme und Bibliotheken 20
9.3 Trennung von Server und Clients 20

10 Konfiguration der bootenden Maschine **20**
10.1 Die zweigeteilte Boot-Prozedur 20
10.2 Wahl der Betriebsart 21
10.3 Einrichtung der Hardware 21
10.4 Debugging 22

11 Konfiguration des Servers **22**

12 Fazit **23**

A VMware auf Diskless Clients **24**
A.1 Überblick 24
A.2 Hardwareanforderungen 25
A.3 Der VMware-"Rechner" 25
A.4 Virtuelle Festplatten 27

A.5 Die Konfigurationsdateien 28
A.6 Zentrale (Hardware-)Konfiguration 29
A.7 Laufwerkskonfiguration 31
A.8 VMware unter Linux 31
A.9 Einsatzbereiche 32
A.10 Windows98 im DXS-Umfeld 34

B Skripten **36**
B.1 Einsatz von Skripten 36
B.2 Einrichtungsskripten 37
B.3 Laufzeitskripten 54

C Eine kurze Installationsanleitung **73**
C.1 Unterstützte Linux-Distributionen 73
C.2 Einrichtung des Client-Dateibaumes 74
C.3 DHCP-Konfigurations-Optionen 75
C.4 Vorbereitung der Client-Rechner 76
C.5 Hardware-Unterstützung und Konfiguration 77
C.6 X11-Konfiguration 78
C.7 Anwendungsprogramme 79

D Weiterführende Information **80**
D.1 Print 80
D.2 Web 80

abstract

Fast alle Arbeitsabläufe, die in der modernen Arbeitswelt vorkommen, sind in irgendeiner Form mit Computern und Computernetzen verknüpft. Damit verbunden ist ein Wachstum bei der Ausdehnung von Rechnerarbeitsplätzen, wodurch die Aufgabe des Betriebes dieser Netze immer umfangreicher wird. Neben Personal, das Wartungs- und Routinearbeiten an Rechnern, ihren Komponenten und im Netzwerk durchführt, erfordert der Betrieb größerer IT-Infrastrukturen eine ständig steigende Anzahl von Experten. Einen entscheidenden Ansatz zur Vereinheitlichung und Automatisierung des Rechnerbetriebes bilden Thin-Clients. Dieser Rechnertyp versucht durch Reduktion der Hardware und durch Zentralisierung im Betrieb Kosten in mehreren Dimensionen zu reduzieren. Wesentlich befördert werden kann dieser Ansatz durch die Verwendung des Betriebssystems Linux. In diesem Vortrag werden die grundsätzlichen Ideen der Linux Diskless Clients vorgestellt. Hierzu zählt die Auswahl der geeigneten Hardware und darauf aufbauende Bootlösungen, die Anforderungen an den Server und spezielle Designs für das Dateisystem der Diskless Clients. Um den Aufwand für mögliche Anwender von Diskless Clients gering zu halten, wurden eine Reihe von Skripten entwickelt, welche die meisten Einrichtungsaufgaben automatisch erledigen. Ergänzt wird das Projekt um angepasste Bootskripte, die Rücksicht auf die besondere Konstellation der verfügbaren Dienste und des nur-lesbaren Dateisystems nehmen. Die Konfiguration der einzelnen Maschinen erfolgt möglichst automatisch, was die Einbindung der Hardware-Ressourcen anbetrifft. Die Betriebsart, ob z.B. die Maschine in eine grafische Oberfläche bootet, einen Chooser oder grafisches Login anbietet oder direkt VMware mit einem bestimmten Betriebssystem-Image startet, wird zentral mittels DHCP vorgegeben. Dieses versorgt den Client nicht nur mit allen entscheidenden Server-IPs, sondern auch mit weiteren Daten zu dessen Betriebsmodus. Die Hardware-Auswahl für das vorgestellte Projekt wird immer einfacher: Linux unterstützt fast alles, was derzeit am Markt verfügbar ist, auch komfotable Ergänzungen, wie USB und Firewire stellen keine Probleme mehr dar. Gebootet werden kann entweder mittels PXE und Syslinux unter Verwendung von DHCP und TFTP oder mittels Etherboot unter Einsatz von DHCP und NFS. Beide Methoden werden gegenübergestellt und Betrachtungen zu Vor- und Nachteilen angestellt. Einiges Augenmerk ist dem Bootvorgang und der möglichst automatischen Einrichtung der Hardware zu widmen. Nachdem eine solide Betriebsplattform geschaffen wurde, wird ein Überblick über Einsatzszenarien für verschiedene Betriebsmodelle und Arbeitsoberflächen gegeben. Eine besondere Rolle spielt hier der Einsatz von VMware: Gemeinsam mit der Virtuellen Maschine läßt sich z.B. Windows98 für Kurse und So ftwarebenutzung fast sicher und optimal administrieren.

this page was left blank intentionally too

1 Einleitung

Seit über einem Jahrzehnt gibt es stabile Abrechnungssysteme für Arztpraxen unter UNIX. Eines dieser Systeme ist DAVID. Es wird seit 1996 unter dem Betriebssystem LINUX eingesetzt. Dabei wurden die textorientierte Stabilität mit den immer weiter entwickelten Möglichkeiten der grafischen LINUX-Oberflächen zusätzlich erweitert. Dadurch können die Anwender sowohl grafisch (auch mit der vorhandenen alten Hardware), wie weiterhin auch rein textorientiert arbeiten.

Etliche Neu- und Weiterentwicklungen in der Netzwerkstruktur sorgen dafür, dass DAVID nicht nur ein Programm unter LINUX ist, sondern auch die speziellen Möglichkeiten der LINUX-Welt nutzt. Neben den altbewährten Textterminals können als Arbeitsplätze natürlich auch Windowsrechner und überhaupt Rechner mit allen Betriebssystemen eingesetzt werden.

Die absolute Krönung ist der lautlose Diskless-Arbeitsplatz, auf dem alle Betriebssystem-Welten dargestellt und gewechselt werden können. Nachfolgend wird ein kurzer Erfahrungsbericht geliefert, der von einem Überblick zum aktuellen Stand dieser Lösung gefolgt wird. In diesem Berichtsteil werden die grundsätzlichen Ideen der Linux Diskless Clients vorgestellt. Zur Illustration werden einige Beispiele und Skripten im Anhang dieses Überblicks präsentiert. Ein eigener Abschnitt widmet sich dem Einsatz von VMware unter Linux allgemein und auf Diskless X-Stations (DXS) im speziellen.

1.1 Ein Erfahrungsbericht

Ein Erfahrungsbericht über den Einsatz von Linux in Arztpraxen gibt das Interview mit Dr.Peter Wolf über den Ablauf der DAVID-X Montage in Mosbach mit der Redaktion der DAVID-X-line:

Redaktion DAVID-X-line Herr Dr. Wolf, Sie hatten vor einiger Zeit den bisherigen Arbeitsablauf in Ihrer Praxis umgestalten wollen. Wie liefen die Arbeiten mit Ihrer Praxis-EDV denn vorher ab ?

Dr. Peter Wolf Wir benutzten DAVID bereits seit vielen Jahren rein textorientiert und hatten bereits Arbeitsstationen, die vom David-Server aus über das Netz gestartet wurden. Dabei habe ich die Stabilität eines LINUX-Systems am Server und an Arbeitsplätzen schätzen gelernt.

Redaktion DAVID-X-line Was wollten Sie am Arbeitsablauf ändern und wie wurde dies verwirklicht?

Dr. Peter Wolf Es sollte möglich sein, an allen Arbeitsstationen genau so zu arbeiten, wie dies auch am Hauptrechner möglich ist. Mehrere textorientierte Sessions ALT F1-F5 und die grafische Session auf ALT-F7 mit ihrerseits bis zu 16 möglichen Arbeitsflächen sind für ein strukturiertes Arbeiten an allen Arbeitsplätzen sehr sinnvoll. Dies wurde an allen Arbeitsplätzen möglich gmacht. Beim Anschalten der Arbeitsplätze, die übrigens ohne Lüfter und Platte auskommen können, wird das "Betriebssystem" über das Netz geladen.

Redaktion DAVID-X-line Welchen Vorteil bringt es für die Praxis, wenn ein Arbeitsplatz "vom Netz gestartet" wird?

Dr. Peter Wolf Nun, dies bringt den gleichen Sicherheitsvorteil, wie er für die reine textorientierte Bedienung von DAVID seit vielen Jahren beim Starten von Terminals, (z.B. Wyse, Dorio usw.) selbstverständlich ist. Alle Informationen, die ein Arbeitsplatz-Rechner benötigt, liegen zentral auf dem DAVID-Server und können dort auch zentral gewartet und gepflegt werden. Dann kann durch das Fehlen einer Festplatte oder sonst bei Arbeitsplatz-Rechnern notwendigen Lüftern, der Arbeitsplatz völlig lautlos bleiben und trotzdem alle grafischen Funktionen, ausreichend schnell dargestellt werden. Auch der Preis für einen Arbeitsplatz ist bei der Anschaffung und, wegen des Fehlens beweglicher Teile, wesentlich günstiger.

Redaktion DAVID-X-line Dann wollten Sie doch an einer Stelle der Praxis einen Windowsrechner behalten, an dem Sie einige spezielle Windowsprogramme (wie für Scanner, Fotoapparat, usw.) laufen haben und standen vor dem Problem, dass Sie also entweder laute Windowsrechner an jedem Abeitsplatz einsetzen müssen oder aber einen Windowsrechner irgendwie "fernbedienen" können. Wie wurde diese Forderung erfüllt?

Dr. Peter Wolf Nun, da wurde eine Lösung gefunden, die mich schlicht weg begeistert hat. Der Windows-Bildschirm wird über ein Programm auf allen Linuxarbeitsplätzen auf einer der 16 grafischen Arbeitsflächen ganz einfach vervielfältigt. Wenn ich von Arbeitsplatz zu Arbeitsplatz wechsele kann ich problemlos überall den Windowsrechner quasi fernbedienen. Dabei kann ich mit Passwort auch festlegen, ob der Bildschirm sichtbar ist oder nicht. Dann gibt es den riesigen Vorteil, daß ich in diesen Rechner eine CD einlegen und genau auf diese CD dann von allen Plätzen aus zugreifen kann.

Redaktion DAVID-X-line Dies hört sich nach einer ziemlich teuren Lösung an. Hat dieses Programm sehr viel gekostet?

Dr. Peter Wolf Dies hatte ich auch zuerst vermutet, aber Familie Heggen (Geschäftsleitung der Multi-Data GmbH) hat mir bestätigt, dass eine solche Möglichkeit ganz normal bei den kostenlosen LINUX-Programmen enthalten ist und keine weitere LIZENZ neben DAVID-X benötigt. Nur die Anpassungen des Technikers vor Ort, damit ich mich nicht selber um die Einrichtung eines solchen Programms kümmern muss und damit DAVID-X auf allen Plätzen optimal angepasst wird, hat den Technikerlohn und eine angemessene "Werkstattpauschale" gekostet.

Redaktion DAVID-X-line Gibt es denn keine LIZENZ-Probleme wegen der Windows-LIZENZ, die Sie ja eigentlich auf vielen Arbeitsplätzen nun nutzbar ist, aber doch nur für einen Platz gilt?

Dr. Peter Wolf Da ich bei dieser Lösung immer nur an dem Platz, wo ich mich gerade befinde, mit dem Windowsfenster arbeiten kann, nutze ich also die Windowslizenz auch nur gleichzeitig einmal.

Redaktion DAVID-X-line Nun hat Multi-Data ja den Patientenordner, dies ist ja ein Verzeichnis, welch es automatisch für jeden einzelnen Patienten erstellt wird, seit einiger Zeit eingeführt. Wie waren Sie damit zufrieden und wie sieht das Handling aus? Auch die Darstellung der gespeicherten Dokumente und Bilder in den einzelnen Patientenordnern wollten Sie übersichtlicher haben. Wie wurde dies verwirklicht?

Dr. Peter Wolf Beim Aufruf eines Patienten öffnet sich auf einer weiteren Arbeitsfläche ein Fenster in dem alle Dateien (Dokumente, Bilder, Befunde usw.) aufgelistet sind. Diese Darstellungsmöglichkeit wollte ich mit weiteren Unterordnern unterteilt haben und diese Einzelheiten konnte ich in einem Gespräch mit Fam. Heggen vor Ort klären. Dabei wurden die diskutierten Möglichkeiten direkt programmiert, getestet und verwirklicht.

Redaktion DAVID-X-line Soll dies bedeuten, dass Ihre Betreuungsfirma immer vor Ort sein muß, um weitere Ergänzungen und Änderungen an Ihrem speziellen EDV-Arbeitsablauf zu realisieren?

Dr. Peter Wolf Auf keinen Fall! Da hat mir Herr Heggen angeboten und verwirklicht, dass solche weiteren Programmänderungen per Fernwartung durchgeführt werden. Nach Vereinbarung gebe ich die Fernwartung frei und DATA-VITAL oder Multi-Data kann mir dann die gewünschten Änderungen oder auch Schulungen über eine direkte sichere ISDN-Leitung ermöglichen. Dies läuft genau so einfach ab, wie das "Abholen der Labordaten" per ISDN. Dabei kann ich auf dem Bildschirm verfolgen, welche Arbeiten der Fernwartungsrechner durchführt.

Redaktion DAVID-X-line Was hat diese Lösung für Sie zusätzlich denn gekostet?

Dr. Peter Wolf Auch dies wurde von Multi-Data mit der "Installationspauschale" und der normalen Arbeit vor Ort ohne zusätzliche Programmkosten, wie man sie von der Windowswelt her kennt, verwirklicht.

Redaktion DAVID-X-line Nun hat Multi-Data zwar einige "Stützpunkte" in Deutschland verteilt, aber Ihre Kollegen werden natürlich von anderen Händlern betreut. Können diese Praxen denn eine solche Speziallösung auch installiert bekommen ?

Dr. Peter Wolf Herr Heggen hat mir versichert, dass die Firma Multi-Data GmbH mit "allen Händlerkollegen vor Ort" bereitwillig und vertrauensvoll zusammen arbeiten wird, damit die aufwendige Konfigurationsarbeit einer solchen LINUX-Netz-Lösung nicht von jedem Händlerkollegen neu erarbeitet werden muss. Damit kann auch jede Arztpraxis eine solche Installation bekommen.

Redaktion DAVID-X-line Wir danken sehr herzlich für dieses Gespräch und für Ihre Bereitschaft auf unsere Fragen zu antworten.

2 Grundsätzliche Idee der Diskless X-Station

Soll eine größere Zahl von Rechnern mit ähnlichen Aufgabenstellungen betrieben werden, so will man nicht jede Maschine einzeln installieren oder warten müssen. Darüberhinaus möchte der Ansatz erreichen, die Geräte je nach Anwendungszweck möglichst "schlank" [1] halten. So sollen Anschaffungskosten reduziert und die Energieaufnahme und Lärmabstrahlung minimieren werden.

Viele Standard-Anwendungen benötigen nicht Rechen- und Grafikleistung einer aktuellen Workstation. Es geht hier verstärkt um die Ergonomie des Arbeitsplatzes. Ein Recherchesystem sollte in lärmsensiblen Bereichen, wie z.B. Bibliotheken wenig Geräusche erzeugen.

[1] hierher rührt der Name "Thin Client"

Da auf Wechsellaufwerke in den meisten Fällen verzichtet werden kann, erreicht man sehr kompakte Bauformen, die sich durch wenig Platzverbrauch und eine geringe Energieaufnahme auszeichnen.

Die Voraussetzungen für den reibungslosen und performanten Betrieb von Linux Thin-Clients sind mit der Existenz leistungsfähiger Ethernet-Installationen üblicherweise gegeben. Fast alle Computeranwendungen der heutigen Zeit greifen in irgendeiner Form auf einen Netzwerkdienst zurück. Diese reichen von der klassischen Datenbank, den Druck- und Fileserver, den Internet- und Verzeichnisdiensten bis zu organisationsinternen Kommunikationsstrukturen aller Art.

3 Einsatzgebiete von DXS

Es gibt eine breite Palette von Anwendungen für Linux Diskless Clients. Diese beginnen mit dem Linux-Desktop als Arbeitsplatzmaschine im Mehrbenutzerbetrieb mit einer Vielzahl von installierten Applikationen, Authentifizierung und Home-Verzeichnissen. Denkbar sind eine Reihe von Kioskanwendungen ohne Anmeldung mit einer eingeschränkten Anzahl von Applikationen. Dazu zählen Recherchesysteme, wie sie in Bibliotheken und Organisationen vorkommen, Kioskterminals für Präsentationen oder Internetterminals für Veranstaltungen, Kongresse und Messen. Darüberhinaus bilden Linux Diskless-Clients die bequem zu administrierende Plattform für Windows-Application-Terminals (Citrix-Metaframe) oder liefern Kurs- und Schulungssysteme. Selbst für vollwertige Grafikworkstations oder Numbercruns-

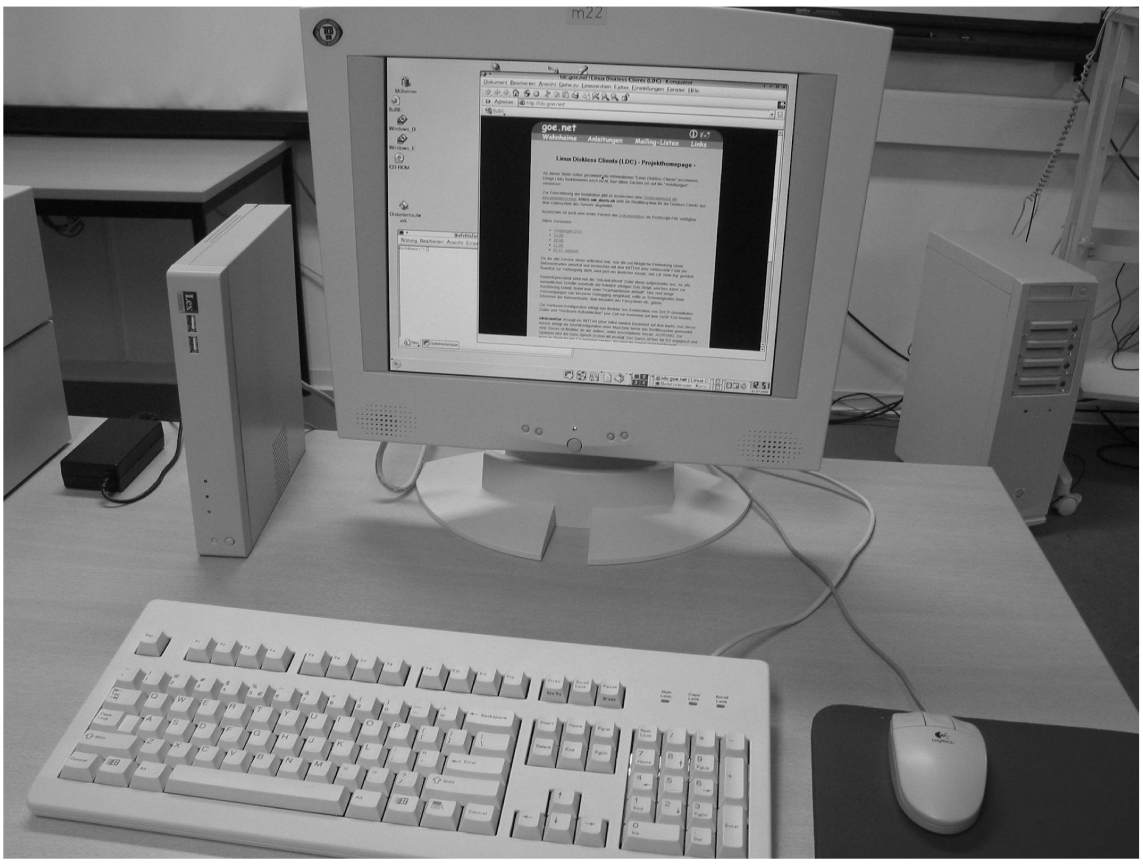


Abbildung 1: Moderne Thin-Client-Hardware auf Basis von PC-Komponenten

her läßt sich mit dem Konzept der Thin-Clients eine ganze Reihe von Kosten sparen, die aus

niedrigeren Anschaffungsaufwendungen und reduziertem Administrationsaufwand resultieren.

4 Designüberlegungen

Es existieren verschiedene Ansätze, um Thin-Clients zu realisieren. Die Spannweite reicht von sehr kompakten Komplettgeräten, die mit stark integrierten Mainboards, Notebookfestplatten und Slim-Line-Laufwerken arbeiten, über Maschinen mit einer kleinen bis größeren Solid-State-Disk, die ein Minimal-Betriebssystem enthalten, bis zu Geräten, die nur über ein kleines Boot-ROM verfügen.

Einen Pluspunkt der mit Festplatte bzw. Solid-State-Disks ausgestatteten Maschinen stellt die größere Netzwerkunabhängigkeit während des Bootvorganges und zum Teil während des späteren Arbeitens an diesem Arbeitsplatz dar. Da zumindest das Betriebssystem bzw. zentrale Applikationen lokal installiert sind, bleiben diese auch bei Störungen des Netzwerkbetriebes nutzbar. Bei einer steigenden Zahl von Maschinen steigt jedoch neben den Hardwarekosten der Aufwand bei Updates und Rekonfigurationen. Weiterhin sollte das Wachstum des Software-Umfanges nicht unterschätzt werden, das recht schnell dazu führt, dass vorhandene Festspeicherkapazitäten nicht mehr ausreichen.

Ein wesentlicher Vorteil der Boot-ROM-Lösung liegt in ihrem geringen Hardware-Aufwand und im Verzicht auf teurere Spezialteile. Mainboards mit Sockeln für Solid-State-Disks oder CF-Anschlüssen kosten mehr als klassische Standardware. Die Wahrscheinlichkeit eines notwendigen Softwareupdates und des Ausfalls wichtiger Komponenten liegt in dieser Ausprägung am niedrigsten. Die Administration erfolgt nach erfolgreicher Hardwareinstallation ausschließlich auf Serverseite. Da dieses Konzept sich am deutlichsten von einer klassischen Betriebssysteminstallation unterscheidet, soll es im folgenden ausführlich vorgestellt werden.

Neben den Dimensionen des Einsatzzweckes und der Softwareausstattung der Thin-Clients wird man sich mit den Aspekten unterschiedlicher Hardware auseinandersetzen müssen. Da selten alle Maschinen zu einem einheitlichen Zeitpunkt beschafft, bzw. selten einheitlich repariert und erweitert werden können, sollte das Betriebssystem der Thin-Clients mit einer gewissen Bandbreite verschiedener Ausstattungen umgehen können. Dieses betrifft klassische Komponenten, wie Grafik-, Sound- und Erweiterungskarten, Mäuse und Monitore, sowie evtl. vorhandene Peripheriegeräte.

5 Protokolle und Technologien

In den nächsten Abschnitten werden nun die zur Umsetzung des Konzeptes notwendigen Netzwerkprotokolle und Technologien erörtert. Dieses betrifft zum einen das Spektrum der zur Auswahl stehenden Boot-(ROM)-Implementationen und zum anderen die notwendigen Netzwerkprotokolle für die Zuweisung von IP- und Konfigurationsdaten, sowie das Client-Dateisystem.

Als Boot-ROM-Implementation wird Etherboot, als schönes Beispiel eines GPL-Tools vorgestellt. Die Pre-Boot-Extension (PXE) ist eine Implementation seitens der Hersteller von Netzwerkhardware, um eine einheitliche Softwareplattform zum LAN-basierten Rechnerstart anzubieten. Beide verwenden zur Beschaffung der notwendigen IP-Grundkonfiguration das Dynamic Host Control Protocol (DHCP), welches auch im weiteren Startvorgang der linuxbasierten Thin-Clients eingesetzt wird. Zur Übertragung des Betriebssystemkerns verwendet man das Trivial File Transfer Protocol (TFTP), wobei Etherboot alternativ dazu das Network File System (NFS) verwenden kann. Dieses kommt überlicherweise anschließend als Root-Filesystem der Clients zum Einsatz. Neben dem Netzwerkdateisystem benötigt man

meistens noch ein RAM-Filesystem, in welchem nichtstatische Daten abgelegt werden können. Die Wahl fiel hier auf das TEMPFS, wobei auch das einfachere RAMFS[2] oder die inzwischen in die Jahre gekommene RAMDISK einer fixierten Größe zum Einsatz kommen können.

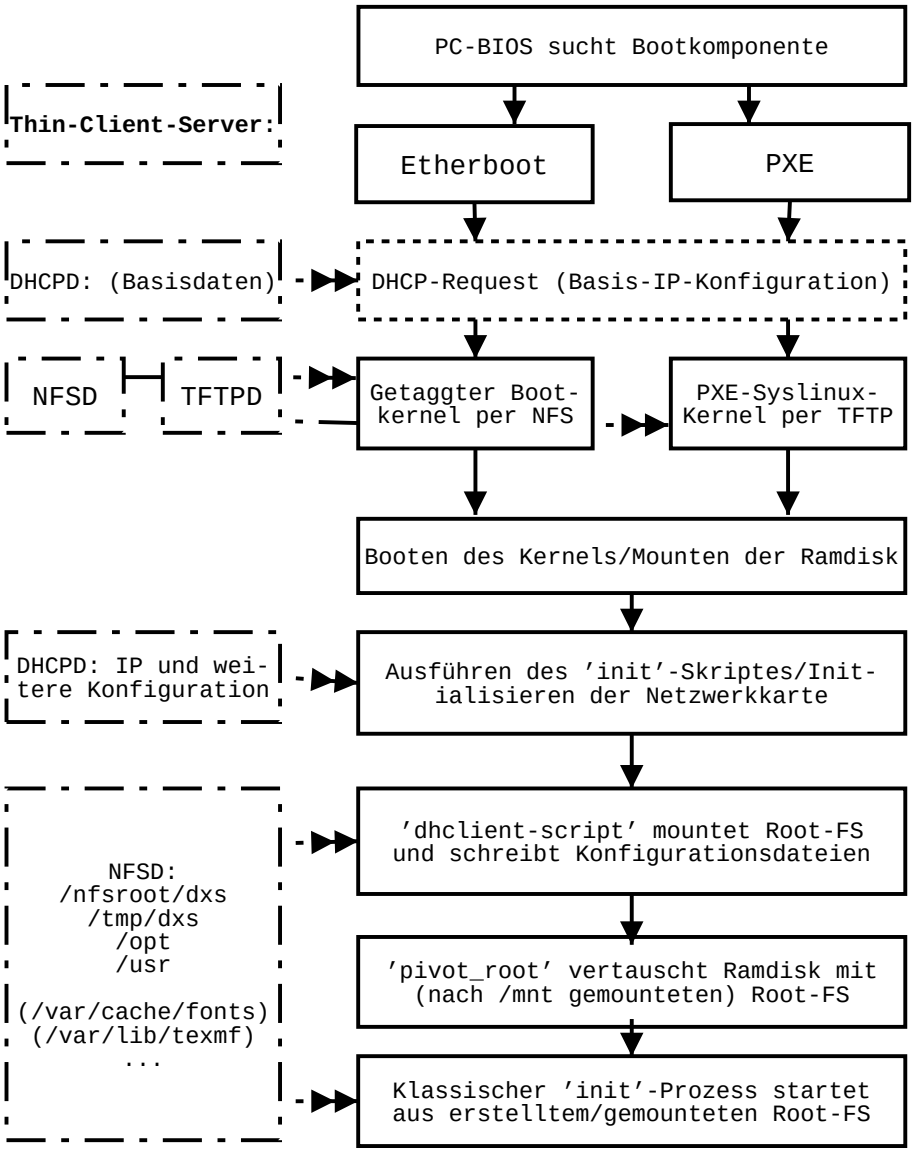


Abbildung 2: Bootablauf eines typischen Thin-Clients

Der netzwerkkopierte Bootkernel benötigt spezielle Einstellungen, um in der beschriebenen Umgebung gestartet werden zu können. Etherboot liefert hierzu ein Tool für das sogenannte Kernel-Tagging[3] mit, PXE kann in Verbindung mit Etherboot arbeiten und auf dessen Fähigkeiten zurückgreifen oder zusammen mit dem Syslinux-Bootloader eine alternative Strategie anbieten.

[2] jeweils variabler Größe, welche sich dynamisch an die abgelegte Datenmenge anpaßt

[3] Der Kernel wird mit einer speziellen Startsequenz versehen, damit dieser nach dem Netzwerktransfer mit evtl. Kernelparametern gerstartet werden kann.

6 Die Client-Boot-Software

6.1 Vorüberlegungen

Die nun folgenden Abschnitte beschäftigen sich mit der Frage, wie die Thin-Clients in die Lage versetzt werden können, über ein Netzwerk zu booten. D.h. die Aufgabe der Boot-Software wird darauf beschränkt, den Client nach dem Start mit einer minimalen Netzwerkfähigkeit auszustatten, um alle weiteren Daten von einem oder mehreren Servern beschaffen zu können.

Thin-Clients sollen einfach zu handhaben sein, dabei aber gleichzeitig eine gewisse Auswahl an Hardware unterstützen, da selten eine homogene Hardware-Ausstattung vorausgesetzt werden kann. Die zu verwendende Bootsoftware soll keine teure Spezialhardware erfordern, sondern möglichst die Gegebenheiten der verwendeten PC-Hardware ausnutzen.

Die Bootsoftware muss nach dem Ausschalten des Gerätes wieder verfügbar und automatisch ohne besondere Benutzerinteraktion im Standardbetrieb aktivierbar sein. Dabei können spezielle Anforderungen, wie z.B. Dual-Boot-Lösungen, etwas andere Implementierungen erfordern. Alle Ansätze sollten sich jedoch aus Sicht der Software auf dem Server möglichst identisch verhalten, um den Aufwand spezifischer Anpassungen gering zu halten[4].

6.2 Etherboot

Das freie Boot-Rom-Paket Das Etherboot-Paket, welches unter der GPL im Internet zur Verfügung steht, enthält inzwischen die Treiberimplementierung fast aller gängiger Netzwerkkarten. Es unterstützt vielfältige Optionen, die ein Zusammenspiel mit anderen Boot-Ladern, Dual-Boot-Lösungen, Boot-Menüs erlauben. Die aus dem Etherboot-Paket kompilierten Boot-Images sind von sehr geringem Umfang, damit sie in handelsübliche EPROMs oder Flash-ROMs zum Einsatz auf der Netzwerkkkarte passen. Alternativ kann der Code auch als Extension-ROM dem BIOS des jeweiligen Mainboards mit Spezialtools hinzugefügt werden[5]. Die Boot-Images belegen zwischen 8 kByte und 64 kByte Speicherplatz. Im Code enthalten sind der Treiber für die Netzwerkkarte und die notwendigen Netzwerkprotokolle DHCP und TFTP bzw. NFS[6].

Weiterhin ist es denkbar, Etherboot als ausführbare DOS-Datei zu erzeugen oder direkt in den Bootsektor einer Diskette zu schreiben, was einfaches Testen und Debugging ermöglicht.

Etherboot erlaubt aufgrund seines Lizenzmodells eine beliebige große Anzahl von Installation ohne Gebühren. Durch die Offenlegung des Source-Codes besteht die Aussicht, eigene Erweiterungen vorzunehmen oder spezielle Anpassungen an das eigene Netzwerk vorzusehen.

Durch die Beschränkung auf wenige Aufgaben kann die Fehleranfälligkeit im Code des erzeugten Bootimages als gering eingestuft werden, was aufwändige Updates der jeweils betroffenen Rechner vermeiden hilft. Eine einmal erfolgreich geteste Installation muss jedoch beim Austausch der Netzwerkkarte gegen einen anderen Typ oder bei bestimmten Veränderungen der Konfiguration des DHCP, wie der Verschiebung der Portadressen oder die Benutzung von Vendor-Code-Identifiern, geändert werden.

[4] Dieses beträfe z.B. unterschiedlich aufzubereitende Kernel für die Zusammenarbeit mit Etherboot bzw. PXE und dem Syslinux-Paket.

[5] Mit den DOS-Programmen **cbrom.exe** für Phoenix/AWARD und **amibcp.exe** für AMI-BIOS können die entsprechenden Modifikationen vorgenommen werden.

[6] Der Codeumfang bei der Auswahl von NFS erhöht sich geringfügig gegenüber dem Einsatz von TFTP und ist bei sehr kleinen EPROM-Größen sehr genau abzuschätzen.

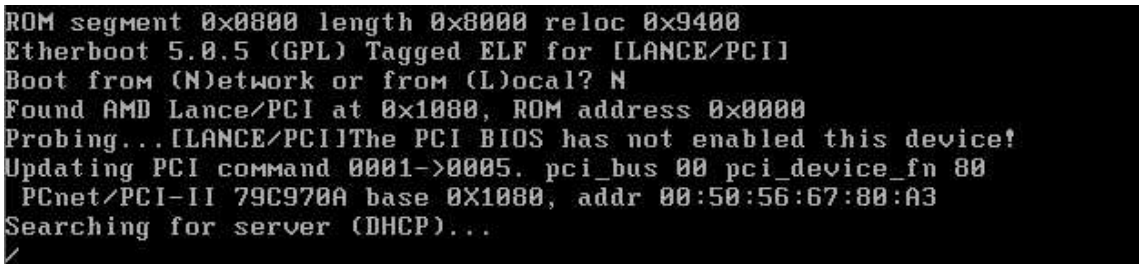


Abbildung 3: Etherboot für Lance/PCI-Netzwerkkarte

Allgemeine Einstellungen Die verschiedenen Optionen für Etherboot lassen sich in der Datei *Config* im Source-Verzeichnis des Quellpaketes vornehmen. Alle zur Verfügung stehenden Optionen sind zu Beginn dieser Datei kurz erläutert. Weitere Ausführungen entnimmt man der umfangreichen Dokumentation. Die Aus- bzw. Abwahl einzelner Optionen entscheidet über den Umfang des später erzeugten ROM-Images[7].

In der Konfigurationsdatei wird bestimmt, auf welche Weise das Kernelimage geladen werden soll. Die aktuellen Etherboot-Implementierungen erlauben zum Laden des Bootkernels anstelle von TFTP alternativ das Network Filesystem zu verwenden, da dieses anschließend für das Rootfilesystem zum Einsatz kommt. So kann auf Serverseite ein Dienst reduziert werden, womit sich die Ausfallwahrscheinlichkeit verringern, sowie die Systemsicherheit erhöhen läßt. Der Einbau der NFS-Unterstützung vergrößert den erzeugten Code um einige Kilobyte.

```
# For prompting ...
CFLAGS32+=     -DMOTD -DIMAGE_MENU
[...]
# Change download protocol to NFS, default is TFTP
CFLAGS32+=      -DDOWNLOAD_PROTO_NFS
# For prompting and default on timeout
CFLAGS32+=     -DASK_BOOT=3 -DANS_DEFAULT=ANS_NETWORK
[...]
# Enabling this makes the boot ROM require a Vendor Class Identifier
# of "Etherboot" in the Vendor Encapsulated Options
CFLAGS32+=      -DREQUIRE_VCI_ETHERBOOT
```

Möchte man die Meldung zur Auswahl der Boot-Art (Starten aus dem Netzwerk oder von einem lokalen Device) modifizieren, so geschieht dieses in der Datei *etherboot.h* welche im selben Verzeichnis wie die Konfigurationsdatei liegt. Soll Etherboot sich mit einem alternativen "Vendor-Code-Identifier", dem String zur Identifikation gegenüber dem DHCP, melden, ist die Datei *main.c* anzupassen[8]. Zum Wirksamwerden letztbeschriebener Optionen ist jedoch das Einschalten in der zentralen Konfigurationsdatei Bedingung.

Kompilation Zur Erzeugung aller **.rom*-Images genügt der einfache Aufruf von **make** im Quellverzeichnis. Diese Images liegen anschließend im Unterverzeichnis *bin32* bereit. Zum Testen der generierten Boot-ROM-Abbilder empfiehlt es sich, diese zuerst mit einer Bootdiskette zu testen. Einzelne Diskettendateien werden direkt durch den Aufruf von **make bin32/name_der_netzwerkkarte.fd0** erzeugt und auf die im Laufwerk befindliche

[7] EPROM-Chips müssen Größen von 128 kbit bis 512 kbit aufweisen, bzw. das BIOS-Flash-ROM über entsprechend freien Platz verfügen. Alle Einstellungen erfolgen in der Konfigurationsdatei von Etherboot: *src/Config*.

[8] Voreingestellt ist der VCI-String "Etherboot".

Diskette geschrieben. Der Unterschied zum später eingesetzten ROM-Image liegt im vorgeschalteten Disketten-Bootheader.

Das Generieren von PXE-Images, welche durch die auf Clients evtl. bereits vorhandene Bootsoftware geladen werden, geschieht auf analoge Weise. **make bin32/name_der_netzwerkkarte.pxe** erzeugt eine entsprechende Datei. Dieses kann direkt in Zusammenarbeit mit PXE getestet werden, da eine Festinstallation auf dem Client unnötig ist.

"mknbi(-linux)" Das Perl-Skript **mknbi-linux** erlaubt neben der Erstellung von Bootimages für Linux auch die Unterstützung weiterer Betriebssysteme, z.B. DOS. Die Manualpage kann mit "man mknbi" eingesehen werden. Das Kerneltagging geschieht z.B. mit: **mknbi -o bootimg -d /nfsroot/dxt -i rom kernelimage [ramdiskimage]**. Die Optionen geben nacheinander das Kernelfile, das Outputfile (der dann netzbootfähige Kernel) und das NFS-Root an. Die letzte Option übergibt dem Kernel die Information, dass er seine IP-Konfiguration ber die DHCP/BOOTP-Anfrage des Boot-ROM's bezieht. Im beschriebenen Projekt wird diese Option nicht benötigt, da die DHCP-Anfrage aus der INITTAR-Umgebung heraus erfolgt.

Multiboot-Anpassungen Interessant sind die einschaltbaren Anpassungen von Etherboot an Multibootumgebungen, die neben dem klassischen Start aus dem Netzwerk auch Optionen zum Booten von Festplatte, Diskette oder CD-ROM-Laufwerk bereit stellen können. Dieses kann direkt vor dem Start aus dem Netzwerk geschehen. Hier wird die Unabhängigkeit von der Verfügbarkeit eines Bootservers sichergestellt. Die Flexibilität späterer Anpassungen ist jedoch begrenzt, da dann der Bootcode eines jeden Clients ausgetauscht werden muss. Begibt man sich in die Abhängigkeit der Verfügbarkeit der Bootserver, läßt sich über diese eine ausgefeilte Bootmenu-Struktur verwirklichen und bei Bedarf dynamisch mittels DHCP- oder BOOTP-Server anpassen. Diese Veränderungen ziehen dann kein Updatebedarf auf Clientseite nach sich. Die Kombination beider Varianten macht keine Probleme, könnte aber am Ende zur Verwirrung des Anwenders führen, da dann mehrfache Auswahlen präsentiert werden.

6.3 PXE

Intels PXE[9] stellt eine weitere Möglichkeit dar, über das Netz eine festplattenlose Maschine zu starten. Mittels DHCP/TFTP wird ein PXE-Image geladen, welches dann weitere Bootfunktionalität zur Verfügung stellt. Das Syslinux-Paket[10] enthält neben anderen Bootfunktionen auch die Unterstützung für PXE. Dieses ist dann in der Lage, spezielle Linux-Kernel per TFTP zu booten und zu starten.

Weiterhin ist es vorstellbar neben Kernel-Images Bootsektoren anderer Betriebssysteme zur Auswahl zu stellen und damit ein netzgesteuertes Multiboot zu erlauben.

Die Preboot Extension läßt sich inzwischen auch gemeinsam mit Etherboot verwenden. Es wird in diesem Fall ein Kettenstart ausgeführt, bei dem zuerst PXE aktiviert wird, ehe dieses ein auf PXE zugeschnittenes Etherboot lädt, das wiederum den weiteren Bootvorgang durchführt. Diese Kombination erweist sich dann als hilfreich, wenn PXE bereits vorhanden und ein Eingriff in die bestehende Hard- und Software nicht erwünscht ist. Ein weiterer Vorteil liegt hier in der Zentralisierbarkeit der Bootkonfiguration, da bei Veränderungen und Updates kein Austauschen der Bootsoftware auf den Clientsystemen mehr erfolgen muss.

[9] Die PXE-Option ist im BIOS einiger Motherboards mit onboard Netzwerkkarte und z.B. in der 3COM 905C enthalten.

[10] Syslinux stellt eine Art erweitertes **loadlin** dar

Dem steht jedoch nun ein etwas umfangreicherer Bootvorgang mit geringeren Eingriffsmöglichkeiten entgegen. Die beschriebene Anwendung setzt jedoch einen aufwändiger konfigurierten DHCP-Server voraus, da er zuerst die PXE-Anfrage und anschließend Etherboot mit den korrekten Daten versorgen muss.

6.4 Syslinux und PXE

PXE-Linux von Peter Anvin wird mit dem Syslinux Paket verteilt. Es stellt einen Second Stage Boot Loader bereit, welche mit PXE zusammenarbeitet, um einen Linux-Kernel mit evtl. Optionen sowie einer Ramdisk mittels TFTP zu laden.

PXE-Linux verwendet einen etwas anderen Ansatz, um die notwendigen Kernel-Boot-Infor-mationen zu beziehen: Die Parameter werden aus einer Datei im Verzeichnis *pxelinux.cfg/* beschafft, deren Namen sich aus der hexadezimalen Repräsentation der IP-Adresse des Clients zusammensetzt, vom Server bezogen. Sollte eine solche Datei nicht existieren, versucht PXE-Linux sukzessive von rechts beginnend Zeichen zu reduzieren und erneut zu matchen. Sollte keine geeignete Datei auf diesem Wege gefunden werden, kann man generell eine Default-Konfigurationsdatei bereit stellen.

7 DHCP - Das zentrale Konfigurationswerkzeug

7.1 Überblick

DHCP, das "Dynamic Host Control Protocol", ist ein UDP-basiertes Netzwerkprotokoll, mittels dessen grundlegende Daten zur Konfiguration eines Clientsystems übertragen werden können. Es arbeitet auf Port 67 zur Anfrage an den Server und auf Port 68 zur Rückantwort an den Client. Dieses Protokoll läßt sich nur sinnvoll in broadcastfähigen Netzen (z.B. Ethernet bzw. TokenRing) einsetzen. Es ist nicht, bzw. nur über den Umweg eines DHCP-Gateway-Servers routebar.

Das Internet-Software-Consortium entwickelt eine Beispielimplementation des DHCP. Der Sourcecode liegt in einer Linuxanpassung vor, so dass eine einfache Übersetzung für die Zielplattform erfolgen kann.

DHCP kann aufgrund seiner Flexibilität zu einem zentralen Konfigurationstool für viele Netzwerk- und Einrichtungsparameter werden. Man kann versuchen, mittels dieses Dienstes alle relevanten Informationen zum Betrieb von Thin Clients zu transferieren. Neben den klassischen Parametern wie Hostname, IP-Adresse, Netzmaske und Gateway, zählt dazu eine Reihe von Server-IP's: X-Display-, Time-, Swap-, NIS-, Druck-Server etc. Weitere Einträge können dem Aufbau von Menü- und "Message of the day"-Optionen für die Boot-ROM-Software Etherboot dienen.

Im Weiteren bezieht sich daher die Beschreibung auf die DHCP-Implementierung des ISC. Diese liegt inzwischen in der dritten Version vor und unterstützt eine ganze Reihe neuer Eigenschaften, wie die Unterstützung von Dynamischen DNS oder Vendor Code Identifier.

7.2 Server

Konfiguriert wird der DHCP-Dämon (**dhcpd**) mittels */etc/dhcpd.conf*. Es stehen etliche weitere Einstellungen zur Verfügung; diese können mit **man dhcp-options** angezeigt werden.

Die DHCP-Konfigurationsdatei hängt in ihrer Ausgestaltung vom verfolgten Einsatzziel ab: So werden für Diskless X-Stations bzw. Linux-Workstations vielleicht eine Reihe zusätzlicher Daten übertragen, die für den Betrieb von Windowsmaschinen nicht relevant sind.

Entsprechende benutzerdefinierte Optionstrings werden eingeführt, um bestimmte Textfelder[11] z.B. zur Konfiguration des X-Servers zu kopieren.

Es gibt eine Reihe von Möglichkeiten die */etc/dhcpd.conf* zu strukturieren. Zusätzlich zum "subnet"-Statement hilft das "group"-Statement dabei, neben der Unterteilung nach Subnetzen, Konfigurationsblöcke mit gleichen Parametern zusammenzufassen. So muss nicht für jeden Host die gesamte Palette wiederholt werden.
Eine Beispielkonfiguration könnte wie folgt aussehen:

```
        option domain-name "subdomain.domain.local second.domain";
        filename "/nfsroot/bootimg";
        use-host-decl-names on;
        default-lease-time 72000;
        max-lease-time 144000;
        subnet 10.16.60.0 netmask 255.255.252.0 {
        option domain-name-servers 10.16.60.21, 10.16.60.100;
        option ntp-servers ntps1.domain.local, ntps2.domain.local;
        option font-servers fontserver.domain.local;
        option x-display-manager x1.domain.local, x2.domain.local;
        # option netbios-name-servers 10.16.63.252;
        option routers 10.16.63.254;
        option broadcast-address 10.16.63.255;
        # range 10.16.60.201 10.16.60.253;
        }
        group {
   option lpr-servers 10.16.60.2;
   host dxs02 {
          hardware ethernet 00:00:1C:D2:87:DF;
          fixed-address 10.16.60.64; }
        [...]
```

7.3 DHCP-Standardoptionen

:

```
# -- global options --
default-lease-time 160000;
max-lease-time 200000;
use-host-decl-names on;

subnet 192.168.2.0 netmask 255.255.255.192 {
        option broadcast-address 192.168.2.63;
        option domain-name-servers 192.168.2.22;
        option domain-name "test.local, test2.local";
        option nis-domain "chnsmi20";
        option nis-servers 192.168.2.3;
        option log-servers 192.168.2.4;
        option netbios-name-servers 192.168.2.5;
        option routers 192.168.2.1;
#       range 192.168.2.50 192.168.2.62;
```

[11] Variablentyp "string"

```
}
# -- client specific --

host test1 {
        hardware ethernet 00:80:48:C2:DD:6A;
        fixed-address 192.168.2.41; }
host test2 {
        hardware ethernet 00:E0:4C:39:10:21;
        fixed-address 192.168.2.42; }
```

7.4 Benutzerdefinierte Optionen

DHCP bietet die Möglichkeit bestimmte, sogenannte Vendoroptionen aufzunehmen, d.h. es können zusätzliche Optionen zu den bereits vorhandenen definiert werden. Hierfür kann der Codenummernbereich von 128 - 255 verwendet werden. Wenn man umfangreiche Informationen übertragen will, sollte man die Paketgröße des BOOTP-Reply-Pakets über den Standard[12] hinaus erhöhen. Diese Optionen werden zu Beginn der Konfigurationsdateien[13] des **dhcpd** bzw. **dhclient** definiert. Die folgende Liste ist eher als Beispiel denn als vollständige Aufzählung zu verstehen, da für gegebene Aufgaben die notwendigen Felder auch völlig anders aussehen können.

```
# -- lot of information to be transferred --
dhcp-max-message-size 1200;
# -- user defined options --
option o128 code 128            = string;
option o129 code 129            = string;
option menudflts code 160       = string;
option motdline1 code 184       = string;
option menuline1 code 192       = string;
option menuline2 code 193       = string;
option menuline3 code 194       = string;
```

Wobei folgende Variablentypen verwendet werden können: *string, integer, boolean, text, ip-number*. Diese lassen sich auch zu Arrays zusammenfassen. Die Option 128 definiert ein "Magic-Packet", welches die Auswertung von Menu-Optionen für Etherboot einschaltet. Standardwerte für die Menü-Auswahl, d.h. welches Feld nach einem gewissen Timeout gestartet werden soll, werden mit der Option 160 festgelegt. Die Zusammensetzung des Menüs, welches nach dem Kontakt von Etherboot mit dem DHCP-Server angezeigt wird, geschieht durch die Optionen 192 und folgende. Hierbei wird für jede Menü-Zeile ein eigenes Feld benötigt. Mittels der Option 129 sind Parameter zum Kernelstart übermittelbar, die z.B. auch den Root-Verzeichnispfad enthalten . Eine "Message of the day"[14] kann durch das Setzen der Option 184 erfolgen.

7.5 Die Verwendung von Vendor-Code-Identifiern

Vendor-Code-Identifier sind als feste Optionen für DHCP definiert: "vendor-class-identifier" für die Identifizierung des Clients durch den Server und "vendor-encapsulated-options" zur

[12] Die Standardgröße des Bootp-Paketes beträgt 572 Byte. Eine Erhöhung auf z.B. 1024 Byte kann durch "dhcp-max-message-size 1024" erreicht werden.

[13] üblicherweise */etc/dhcpd.conf* für den Server-Dämon sowie */etc/dhclient.conf* für den Client

[14] Hier das Textfeld über der Menu-Auswahl

Identifizierung des Servers seitens des Clients.

Auf diese Weise lassen sich die DHCP-Anfragen verschiedener bootender Rechner voneinander differenzieren, so dass es gelingt an eine Maschine in Abhängigkeit vom anfragenden Client verschiedene Werte für die gleiche Option zurückzuliefern. Dieses ist zwingend notwendig, wenn PXE und Etherboot hintereinander verkettet booten sollen, da PXE zwar die identische IP-Konfiguration erhält, aber anstelle des Kernel-Images das Etherboot-PXE-Image zur Ausführung laden soll. Dieses sieht man an unten stehendem Beispiel: Es werden bestimmte Optionen nur gesetzt bzw. die Default-Option überschrieben, wenn in der Anfrage des Clients ein bestimmter String identifiziert werden kann.
Die Interaktion mit PXE und Etherboot mittels VCI ist im folgenden Beispiel demonstriert.

```
# -- vendor identifier dependend settings --
class "Etherboot" {
  match if substring (option vendor-class-identifier,0,9)="Etherboot";
  option o128 E4:45:74:68:00:00;
  option motdline1 = "Welcome to Testlabs";
  option vendor-encapsulated-options 3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
class "PXEClient:" {
  match if substring (option vendor-class-identifier,0,10)="PXEClient:";
  filename "/nfsroot/dxs/boot/eb-3c905c.pxe";
}
[...]
```

7.6 DHCP-Client

Der Dienst zur clientseitigen Konfiguration, d.h. den dynamischen Bezug einer IP-Adresse seitens der Linuxmaschine, wird über das Kommando **dhclient** zur Verfügung gestellt. Die Konfiguration sollte passend zum DHCP-Server erfolgen. Die Einstellungsdatei lautet */etc/dhclient.conf*:

```
# /etc/dhclient.conf
#
send dhcp-max-message-size 1200;
send dhcp-lease-time 18000;
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, host-name;
require subnet-mask, domain-name-servers;
timeout 60;
retry 60;
reboot 10;
select-timeout 5;
initial-interval 2;
script "/sbin/dhclient-script";
```

Nachdem **dhclient** die DHCP-Informationen vom Server bezogen hat, geschieht die Eintragung bzw. Anwendung dieser mittels **dhclient-script**, welches im Anhang als Shell-Skript beispielhaft angeführt ist. **dhclient-script** wird von **dhclient** mittels Temporärskript in */tmp* mit allen erhaltenen Variablen gestartet. Diese werden als Shell-Variablen in der aufrufenden Kommandozeile übergeben.

8 Dateisystem und Softwareverwaltung

8.1 Aufteilung und Dateisysteme

Die Einrichtung eines Filesystems, speziell des Root-Filesystems eines Thin-Clients ohne eigenen Festspeicher, weicht zwangsläufig von der gewohnten Festplatteninstallation ab. Dabei gibt es eine Reihe von Designaspekten, die beachtet werden sollten. Der Aufbau des Dateisystem der freien Unix-Derivate unterwirft sich dem File System Hierarchy Standard, welcher eine Einteilung der Dateien nach lokal bzw. verteilbar und statisch bzw. dynamisch vornimmt.

Erklärtes Ziel der Einrichtung des Dateisystems der Thin-Clients ist es, eine große Zahl von Maschinen unabhängig von ihrer späteren Funktion oder ihrer Ausstattung aus einem einheitlichen Verzeichnis bedienbar zu machen. Es spart Festspeicherplatz, welcher angesichts heutiger Festplattengrößen nicht mehr der Mangelfaktor ist, aber bei sehr vielen Geräten zu einem Administrationsproblem wird. Zusätzlich wirkt sich ein einheitliches Verzeichnis für alle Clients günstig auf das Caching-Verhalten des Servers aus. Häufig angeforderte Datenblöcke können so zügig aus dem Arbeitsspeicher bedient werden.

Eine weitere Frage ergibt sich aus dem Typ des Servers. Verfügt diese Maschine über dasselbe Betriebssystem auf derselben Prozessorarchitektur, lassen sich direkt Teile des Serverfilesystems einbinden. Man wird aber darauf verzichten, das komplette Rootfilesystem freizugeben, da neben sicherheitsrelevanten Einschränkungen bestimmte Dateien und Verzeichnisse ausschließlich von einem Betriebssystem selbst benutzt werden dürfen[15]. Die Verzeichnisse mit den Standardapplikationen */opt* und */usr* lassen sich generell unproblematisch freigeben. Auf den Client-Maschinen genügt hier ein lesender Zugriff. Größerer Aufwand wird für das Rootverzeichnis, */etc* und */var* einzuplanen sein. Ein Spezialverzeichnis wie */dev* muss nicht exportiert werden, wenn man auf den Device Filesystem Daemon setzt. Sonst muss dieses, wie auch */etc* und */var* in der jeweiligen Ramdisk des Clients angesiedelt sein.

Unter diesen Voraussetzungen sieht eine Aufteilung für den Export der notwendigen Verzeichnisse wie nachstehend für NFS (*/etc/exports*) aus:

```
# Basis-Verzeichnis (Rootfilesystem der Clients)
/nfsroot/dxs     172.20.156.0/255.255.255.0(ro,no_root_squash)
# Erweiterung des /tmp fuer bestimmte Userbeduerfnisse (viel Platz!)
/tmp/dxs         172.20.156.0/255.255.255.0(rw,no_root_squash)
# Hauptverzeichnisse fuer Applikationen
/opt             172.20.156.0/255.255.255.0(ro)
/usr             172.20.156.0/255.255.255.0(ro)
# Zusätzliche Bereiche fuer den Betrieb von LaTeX
/var/lib/texmf   172.20.156.0/255.255.255.0(ro)
/var/cache/fonts 172.20.156.0/255.255.255.0(rw)
```

Die Installation neuer Software beschränkt sich somit in den meisten Fällen auf einen reinen serverseitigen Prozess. Das erleichtert die übliche Administration sowie Softwareupdates erheblich. Nur evtl. Anpassungen im Bereich von */etc* und */var* (im obigen Beispiel notwendig für den Betrieb des LaTeX-Textsatzsystems) sind separat für die Clients vorzusehen.

Wählt man einen Server einer anderen Prozessorarchitektur bzw. Softwareplattform, wird man ein einheitliches Root-Filesystem bestehend aus einer Zusammenfassung von */opt*

[15] Konfigurationsdateien, Temporärverzeichnis mit Sockets für XFree86 etc.

und */usr* sowie */nfsroot/dxs* wählen. Änderungen ergeben sich weiterhin im Bereich der Softwareinstallation. Diese kann meist nicht mehr einfach serverseitig vorgenommen werden, sondern muss z.B. über den Umweg eines Client-Systems erfolgen, welches gegenüber den anderen Maschinen über herausgehobene Zugriffsrechte verfügt. Dieser Client könnte alternativ in VMware virtuell zusätzlich auf dem Server laufen.

Jeder beschriebene Fall erreicht jedoch eine zentralisierte Softwareverwaltung, die für eine homogene Arbeitsumgebung sorgt. Die Installation und Verbreitung von OpenSource-Software bereitet in diesem Setup keine Probleme. Jedoch muss bei lizenzpflichtiger Software an entsprechende Vorkehrungen gedacht werden, da eine ungewollte Verbreitung über sehr viele Thin-Clients ohne Schwierigkeiten erfolgt.

Alle Daten, die statisch bereit gestellt werden können, in erster Linie also das Basis-Root-Filesystem (*nfsroot/dxs*) und */opt*, */usr* werden ReadOnly per NFS eingebunden. Hier ließen sich auch andere Netzwerkdateisysteme, wie Samba (SMB)[16] oder AFS (Andrew Filesystem) als Alternative denken. Bereiche des Dateisystems, welche auch beschreibbar sein sollen, benötigen nach wie vor eine gesonderte Behandlung. Dynamische Daten wie Konfigurationsdateien, Logfiles, Sockets und ähnliches werden im Arbeitsspeicher des Clients (TEMPFS, RAMFS oder Ramdisk) abgelegt. Dementsprechend sind Links auf die verschiedenen Bereiche eingeführt. Diese Aufteilung vermeidet Interferenzen zwischen einzelnen Clients: Die gemeinsam genutzten Verzeichnisbäume sind bis auf bestimmte Ausnahmen nur lesbar und damit nicht manipulierbar. Dieses erhöht die Sicherheit des Systems vor Manipulationen und erlaubt eine einfache zentrale Überwachung des Dateisystems.

Weitere Beachtung sollte die Anlage des Minidateisystems erhalten, welches zur Basiskonfiguration der Maschine per Initial-Ramdisk (INITRD) oder Initial-TAR (INITTAR) geladen wird.

8.2 Aufsetzen des Dateisystems

Ausgehend von der Annahme einer identischen Prozessorarchitektur und ähnlicher Softwareausstattung von Client und Server, steht die Aufgabe, aus einem vorliegenden System das Rootfilesystem der Clients zu extrahieren. Das geschieht am besten per Shellskript: Zuerst erzeugt dieses eine komplette Verzeichnisstruktur, in welche entweder Teile des Serverdateisystems kopiert oder verlinkt werden oder an welche während der Initialisierungsphase Teile des Serverdateisystems gemountet werden. Weiterhin beinhaltet es das Kopieren oder Verlinken aller notwendigen Programme und zugehöriger Basisbibliotheken aus den Verzeichnissen */lib*, */bin* und */sbin*. Wird das Client-Rootfilesystem auf einer separaten Partition des Servers abgelegt, werden die Dateien kopiert, im anderen Falle direkt verlinkt (Hardlinks)[17]. Der Vorteil von Hardlinks liegt darin, dass Änderungen im Serverdateisystem sich sofort auch für die Clients auswirken, ein Verhalten, was in fast allen Fällen erwünscht ist. Vorsicht ist jedoch bei Hardlinks geboten, wenn sich durch wesentliche Updates Inodeneinträge ändern.

In fast jedem Bereich ist darauf zu achten, dass spezielle maschinenspezifische Verzeichnisse, wie z.B. */lib/modules*, nicht einfach kopiert, sondern mit den entsprechenden Daten für die Thin-Clients gefüllt werden.

Aufwändiger wird es bei der Entscheidung über die Ablage der Daten im Bereich */var* und */etc*. Bestimmte Maschinen und IP-spezifische Konfigurationsdateien, wie z.B. *hosts*, *resolv.conf*, *HOSTNAME*, *fstab* ... sollten auf jeden Fall in der Ramdisk liegen. Andere Bereiche, wie z.B. */etc/opt*, die durchaus sehr umfangreich ausfallen, wird man wiederum vom

[16] Jedoch sind dann einige Anpassungen notwendig, so dass alle wichtigen Unixdateitypen unterstützt werden

[17] Softlinks sind nicht NFS-transparent!

Server per NFS einbinden (gemeinsam mit dem Rootfilesystem). Diese Daten sind zumeist statisch (und unterscheiden sich nicht von Client zu Client) und würden sehr viel wertvollen Arbeitsspeicher in einer Ramdisk belegen. Bei der konkreten Aufteilung der einzelnen Dateien und Verzeichnisse kommt es immer auf die jeweiligen Erfordernisse an: Starke Reduktion des Ramdiskanteils geht mit einem höheren Aufwand der Aufteilung einher. Im vorgestellten Beispiel wird das Konfigurationsverzeichnis */etc* in den statischen Teil */etc.s* und den Ramdiskteil */RAM/etc* aufgeteilt. Ersterer wird vom Server ReadOnly exportiert. Der Rest ist bereits Bestandteil des INITTAR und wird per Link auf */etc* abgebildet. Dieser Bereich enthält für alle statischen Elemente Links nach */etc.s*.

Da die Einrichtung von Thin-Client kaum mittels klassischer Setup-Tools in Bezug das Einrichten des Dateisystems erfolgen kann, sind diese Aufgaben gesondert realisiert. Die oben beschriebenen Tätigkeiten übernimmt **mk_dxsfs.sh** für die Erstellung des Rootfilesystems.

Einen ähnlich gelagerten Auftrag erfüllt **mkdxsinittar**, welches alle notwendigen Dateien, wie ausführbare Programme, Bibliotheken und Konfigurationsskripten zu einem INITTAR verschnürt und dieses an den Bootkernel anhängt.

8.3 Das INITTAR

Der ganze Aufwand um die InitRD (Initial Ramdisk) ist nicht nur unhandlich, sondern spätestens seit Einführung des TEMPFS auch hochgradig überflüssig, veraltet und unflexibel. Die Verbesserung basiert auf einer einfachen Idee: Mounten eines leeren TEMPFS als Rootfilesystem, Auspacken eines (komprimierten) TAR-Images (das mit Kernel auf dem selben Wege mitgegeben wird, wie zur Zeit die InitRD) und starten des **init**(-Ersatzes).

Die Einrichtung eines INITTAR erleichtert sich gegenüber der alten Ramdisk erheblich. Man ist nicht mehr auf eine fixe Größe festgelegt, welche erst aus Nullen erzeugt, dann loopback gemountet und zum Schluss noch mit einem Dateisystem versehen werden muss. Das Erstellen eines TAR-Files mit anschliessender klassischer Gzip-Komprimierung genügt und die Größenfixierung sowie das Formatieren entfallen. Für das beschriebene Vorgehen wird ein Kernelpatch und die anschliessende Neuübersetzung des Kernels benötigt.

9 Software-Anpassungen

9.1 Kernel

Für Diskless Clients benötigt man meistens einen speziell aufbereiteten und in seinen Funktionen zusammengestellten Kernel. In Abhängigkeit der jeweils eingesetzten oder zum Einsatz geplanten oder erwarteten Hardware müssen die notwendigen Treiber fest eingebunden oder als Module kompiliert werden. Fest eingestellt wird die Unterstützung verschiedener CPU-Typen. Hier einigt man sich entweder auf den kleinsten gemeinsamen Nenner von Funktionen, die alle eingesetzten CPUs unterstützen. Das betrifft besonders das Zusammenspiel der Prozessoren verschiedener Hersteller, wie VIA, Intel und AMD. Lösbar wäre dieses Problem mit der Verwendung angepasster Kernel für jede Architektur, was jedoch den Pflegeaufwand der gesamten Anlage erhöht. Ähnliches gilt für die Speicherkonfiguration: Diese ist jedoch unkritischer, da Ausstattungen, die über 512 MByte in den Clients hinausgehen eher selten sind.

Alle nicht zum Booten in das INITTAR benötigten Kernelkomponenten sind am besten in Module auszulagern. Dieses hält den Kernel schlank und erlaubt bequeme Ergänzungen. Am einfachsten geschieht der Einsatz von neuen Netzwerkkartentypen, in dem das entsprechende zum Kernel passende Modul kompiliert, dem INITTAR hinzugefügt und in der

modules.dep eingetragen wird. Wären alle eingesetzten Netzwerkkarten fest im Kernel verankert, müsste mit jeder Änderung eine Neuübersetzung erfolgen. Genauso kann auf diese Weise auf später hinzukommende neue Peripherie, wie z.B. für USB oder IEEE1394 reagiert werden.

Komplizierter wird die Situation für softwareabhängige Kernelmodule. Für die Benutzung der Rechner-Emulation VMware werden mindestens zwei spezielle Kernelerweiterungen benötigt. Diese müssen jedoch zur eingesetzten Software-Release passen und gegebenenfalls erneut übersetzt werden. Vom Softwarehersteller mitgelieferte Module helfen selten weiter, da sie sich an den Standardkernels der Linuxdistributionen orientieren. Eine ähnliche Situation ergibt sich mit dem Kernelmodul zur Unterstützung der 3D-Treiber der NVidia-Grafikkarten oder dem Kernelmodul für die IPsec-Verschlüsselungssoftware von Cisco.

9.2 Programme und Bibliotheken

Der Einsatz der beschleunigten OpenGL-Bibliotheken für NVidia-Grafikkarten erfordert die Installation der mitgelieferten Biblitotheken, deren geeignete Verlinkung und Einbindung in der *XF86Config*. An dieser Stelle muss auf die Verträglichkeit des Kernelmoduls mit der jeweiligen Bibliothek geachtet werden, da sonst der Xserver nicht funktioniert. Für weitere Grafikkarten sind eventuell andere Anpassungen in der *XF86Config* nötig, um eine korrekte Funktion sicherzustellen. Diese Probleme müssen im Hardwarekonfigurationsskript (**hwsetup**) abgefangen und geeignet gelöst werden.

Werden speziell optimierte C- oder andere Softwarebibliotheken auf dem Server oder einzelnen Clients eingesetzt, so muss sichergestellt sein, dass diese auch auf der Maschine mit der schwächsten CPU lauffähig sind.

Ähnliche Überlegungen treffen für die Software innerhalb des INITTAR und der späteren Root-Umgebung zu. Die verwendeten C-Bibliotheken müssen zueinander passen, damit Kommandos, die aus dem später übers Netzwerk gemounteten Bereich aufgerufen werden, nicht mit geladenen Routinen der im INITTAR verwendeten C-Bibliothek oder anderen kollidieren. Deshalb sollte nach jeder Softwareneuinstallation oder wesentlichen Updates die Diskless-Umgebung neu inittialisiert werden. Weitere Anmerkungen zum Thema findet sich im Anhang zur Installationsanleitung.

9.3 Trennung von Server und Clients

Nach den soeben geschilderten Problemen und den Ausführungen zur Aufteilung des Dateisystems kann es sinnvoll erscheinen, die Dateisysteme von Server und Client komplett zu trennen. Dieses ist zwingend notwendig bei unterschiedlichen Softwarearchitekturen jedoch auch für erhöhte Sicherheitserwägungen sinnvoll. Dann wird auf dem Server nur die minimal notwendige Software, wie DHCP, NFS und falls benötigt TFTP installiert. Die Einrichtung der Client-Software muss dann auf dem Server über eine Change-Root-Umgebung[18] oder ähnliches erfolgen. Es könnte auch ein spezieller Client, z.B. in einer virtuellen Maschine die auf dem Server läuft, zum Einsatz kommen.

10 Konfiguration der bootenden Maschine

10.1 Die zweigeteilte Boot-Prozedur

Die Initialisierung des hier vorgestellten Linux Diskless Clients erfolgt analog zum Kernel-Start neuerer Linuxdistributionen (z.B. ab SuSE 7.3). Vor dem eigentlichen Mounten des

[18] RPM bietet an in Verzeichnisse zu installieren

Rootfilesystems wird eine minimale Ramdisk-Umgebung ausgeführt, welche eine Reihe von Konfigurationsaufgaben bekommt, z.B. das Laden spezieller Kernelmodule für Dateisysteme oder RAID-Festplatten.

Die Vorteile der Zweiteilung der Boot-Prozedur liegen in der Vereinfachung des Kernels und des Debuggings. Nur jene Kernelemente sind fest in diesen eingebunden welche man zum Start benötigt, alle weiteren können bei Bedarf nachgeladen werden. Dieses verschlankt den Kernel erheblich gegenüber einer Ausgabe, die z.B. alle Netzwerkkartentreiber enthält wovon immer nur genau einer zum Einsatz kommt.

Dazu wird anstelle des **init** ein Shellskript mit dem gleichen Namen ausgeführt, welches zuerst das notwendige Netzwerkkartenkernelmodul ermittelt und lädt und anschließend per dhclient IP- und weitere Konfigurationsdaten beschafft. **dhclient-script** übernimmt sodann die IP-Konfiguration, das Mounten und Zusammensetzen des späteren Dateisystems und das Schreiben vieler Konfigurationsdateien.

Nach dem erfolgreichen Lauf von **dhclient-script** übergibt dieses wieder an **init**, welches das Ramdisk-Filesystem nach */RAM* und die gemounteten Root-Filesystemteile auf die oberste Verzeichnisebene nach / umhängt. Zum Schluss wird aus dem neuen Root-Filesystem das klassische **init** gestartet, womit alle Aufgaben der Initialisierung beendet sind.

Zu diesem Zeitpunkt startet die gewohnte Boot-Prozedur mit ihrem Runlevel-System. Die Steuerung erfolgt wie gewohnt mittels Sys-V-Init durch Start-/Stop-Skripten unterhalb des Verzeichnisses */etc/init.d*. Ob bestimmte Dienste oder Programme gestartet werden oder nicht, wird mittels Eintrag in die */etc/rc.config* entschieden, welche wärend der Initialisierungsprozedur entsprechend den durch DHCP beschafften Daten angepasst wird. */etc/init.d/boot* wurde an den Betrieb festplattenloser Systeme angepasst, alle anderen Skripten konnten in den meisten Fällen direkt kopiert werden. Einige entfielen, da die Netzwerk- und Routingkonfiguration sowie das Mounten der NFS-Dateisysteme ja bereits erfolgte.

10.2 Wahl der Betriebsart

Mittels dynamischer Anpassung der */etc/inittab* wird die Betriebsart des Clients im grafischen Modus gesteuert. Die Konfiguration erfolgt durch die DHCP-Daten, die Auswahl des Windowmanagers und dessen evtl. notwendige Konfiguration bzw. den Start bestimmter Programme übernimmt **startgui**.

Im klassischen Betriebsmodus, der Präsentation eines grafischen Logins bzw. eines Hostchoosers, wird der X-Server direkt mittels **init** und */etc/inittab* gestartet. Die Auswahl des Displaymanagers und grafischen Logins kann ebenfalls durch eine DHCP-Variable erfolgen.

10.3 Einrichtung der Hardware

Eines der Ziele dieses Projektes liegt in der Unterstützung einer Reihe von unterschiedlicher Hardware. Damit nicht für jede einzelne Maschine ein eigenes Rootdateisystem mit den entsprechenden Einstellungen generiert werden muss, wurden zwei Wege der Konfiguration eingeführt. Zum einen lassen sich zentrale Vorgaben, wie die Auswahl des notwendigen Grafiktreibers, die Monitorauflösung, Maus oder das Laden bestimmter Kernelmodule mittels DHCP mitteilen. Zum anderen wird eine "automatische Hardwareerkennung" eingefügt, welche auf dem SuSE-Tool **hwinfo** aufsetzt.

Weitere Aufgaben dieses Skriptes umfassen das Setup der Soundkarte und die Einrichtung der Removable Devices, wie Diskettenlaufwerke und CD-Rom. Für letzteres werden die Konfigurationsdateien der Mountpoints und des Automounters */etc/fstab* bzw. */etc/auto.misc* ergänzt und angepaßt.

10.4 Debugging

Auf festplattenlosen Systemen gestaltet sich die Fehlersuche etwas komplizierter; es muss bereits fast alles, insbesondere jedoch die Netzwerkkonfiguration funktionieren, bevor z.B. ein Logservice gestartet werden kann. Eine gute Informationsquelle für auftretende Fehler liegt im Serverlogfile. Wenn man auf dem Thin-Client den Secure-Shell-Daemon (**sshd**) startet, kann ein Teil der Fehlersuche, gerade bei der Konfiguration des XFree86 auch remote erfolgen. Generelle Überlegungen zum ausführlichen Debugging von Diskless Clients finden sich in [9] im Anhang.

Die grundsätzliche Konfiguration des Thin-Clients soll so vollständig wie möglich mittels verschiedenen Einträge in der Konfigurationsdatei des **dhcpd** erfolgen, um einen möglichst einfachen Überblick behalten zu können. Bestimmte host- oder devicespezifische Dateien sollten soweit es geht vermieden werden, da diese bei Anpassungen oder Updates häufig vergessen werden und damit eine nicht seltene Fehlerursache bilden.

Um das Debugging in den ersten Bootschritten zu vereinfachen, sind in den Start- und Konfigurationsskripten Debug-Meldungen eingebaut, welche über die Kernel-Command-Line eingeschaltet werden können. Diese erklären je nach fehlgeschlagenem Befehl, worin Probleme liegen könnten und wie Abhilfe erreicht werden kann (siehe hierzu das Listing des **init**-Skriptes im Anhang). Weitere Informationen lassen sich einer Logdatei entnehmen, die während des Bootvorganges und der Einrichtung der Hard- und Software geschrieben wird: */var/log/boot.log*.

11 Konfiguration des Servers

Mit der Verabschiedung vom Paradigma des autonomen PCs ergeben sich an den Server und das Netzwerk erhöhte Anforderungen. Fällt ein Server aus, ist nicht mehr das Einzelplatzsystem betroffen oder ein Dienst, wie File- und Druckserver nicht mehr verfügbar, sondern im Extremfall stehen alle Diskless Maschinen. Ähnliches trifft auf ein überlastetes oder unzuverlässiges Netzwerk zu. Hierbei sind, wenn nicht gerade der Anschluss des Servers betroffen ist, meistens nur einzelne Maschinen in ihrer Funktion behindert. Diese Probleme bekommt man bei einer sauberen Planung der Anforderungen und durch Serverredundanz in den Griff.

Weitere Aspekte liegen in der Performance einer Anlage bestehend aus Servern und Diskless Clients. Die Server sollten am besten mittels Gigabit-Interface an die unterverteilende Switch angeschlossen werden, wobei sich am anderen Ende durchaus meherere Clients einen 100 Mbit Port teilen können. Der Ausbau des Hauptspeichers gehört zu den zentralen Möglichkeiten die Leistungsfähigkeit zu steigern. Können die Anfragen der Clients bereits aus dem Filesytemcache des Servers[19] beantwortet werden, steigen die Antwortgeschwindigkeit und Datendurchsatz. Bei vielen identischen Zugriffen, z.B. dem simultanen Booten von vielen Clients, kann die Transfergeschwindigkeit noch über der von normalen Systemfestplatten liegen.

Die geplanten Anwendungen bestimmen das Systemdesign mit: Kommen umfangreiche Pakete, wie z.B. OpenOffice zum Einsatz, sollte die Hardware entsprechend leistungsfähig ausgelegt sein. Der Start eines solchen Paketes benötigt einen hohen Peak-Durchsatz des Netzwerkes, um zügig vonstatten zu gehen. Zu unterschätzen sind auch nicht aufwändige grafische Oberflächen wie KDE. Einzelne Module können aufgrund ihrer starken Desktoporientierung Caching- und Checkingmechanismen auf Veränderungen im Dateisystem einsetzen, die sich im Netzwerk von Diskless Clients negativ auswirken. Ein permanentes "Grund-

[19] Der Linuxkernel fängt automatisch an im Hauptspeicher einen Cache von häufig nachgefragten Speicherseiten anzulegen.

rauschen" von bis zu 2 Mbit/s, die KDE-benutzende Clients verursachen, werden unter bestimmten Situationen zum Problem.

12 Fazit

Die Thin-Clients erfordern zur Installation einen gegenüber der klassischen Workstation erhöhten Einrichtungsaufwand, der sich jedoch bei einer schon geringen Zahl gleichartiger Rechnerarbeitsplätze bezahlt macht. Die hier präsentierte Lösung hatte bei ihrer Implementierung die spätere Skalierung im Blick: Ausgehend von der einmal geschaffenen Basis läßt sich mit nur geringem Mehraufwand, der stark unter dem für die einzelne klassische Workstation liegt, ein weit höhere Zahl verwalten.

Bei den meisten hier beispielhaft beschriebenen Skripten spielt die Zahl der tatsächlich verwalteten Systeme keine Rolle. Hier liegen die Beschränkungen eher in den Möglichkeiten der betrieblichen Organisation, der Leistungsfähigkeit des Netzwerkes und der gegebenen Anforderungen.

Thin-Clients mit Solid-State-Disk oder anderem Festspeicher verlagern einige Software und bestimmten Aufwand vom Server weg und erreichen eine gewisse Autonomie. Dadurch erhöht sich jedoch der Kostenaufwand für die nun einzusetzende recht spezielle Hardware, die Anforderungen an die Ausstattung mit bestimmten Hardwareschnittstellen für die Clients und die Pflege des nun wieder lokal verfügbaren Filesystems. Im Zuge allfälliger Updates aufgrund von Sicherheitsanforderungen oder der Implementierung neuer Eigenschaften erhöht sich üblicherweise auch der Speicherplatzaufwand, der von der ursprünglichen Lösung vielleicht nicht mehr bereitgestellt wird. Damit sind Auswirkungen auf den Gesamtaufwand verbunden.

Zuletzt kann durch die Einführung von Thin-Clients die Sicherheitsarchitektur verbessert werden: Da große Teile des Filesystems vom Server nur lesbar zur Verfügung gestellt werden, sind Manipulationen an wichtigen Binärdateien wie Programmen und Bibliotheken erschwert. Außerdem gelingt es nicht mehr so einfach, zusätzliche Programme, Log-Dateien oder Skripten zu verstecken, die Systemleistung blockieren oder Angriffe einleiten. Der Umfang der zu überwachenden Filesysteme sinkt rapide, so dass sich auch aufwändigere Überwachungssysteme anbieten, ohne den Gesamtadministrationsaufwand erheblich zu erhöhen. Das Augenmerk liegt auf den Servern, die in vielen Fällen bereits in Sicherheitsstrukturen eingebunden sind.

In der vorgestellten Betriebslösung verschiebt sich die Sensibilität in Bezug auf Sicherheit und Konfiguration zum Server hin. Systemadministratoren müssen nun verinnerlichen, dass Änderungen am Serverfilesystem oder die Verfügbarkeit von Applikationen sich über den Server hinaus auswirkt. Die Stabilität des gesamten Betriebes hängt nun in höherem Maße von den Servern ab. Dieses betrifft auch Erwägungen zur Systemsicherheit. Einbrüche auf die zentralen Maschinen haben nun weiterreichende Auswirkungen, bis hin zur Korrumpierung bzw. Unbrauchbarkeit der Clients.

Der abgestufte Bootprozess mit seiner Zweiteilung in einen initialisierenden und einen klassischen Teil erlauben vielfältige Anpassungen. So lassen sich verschiedene Netzwerkdateisystemtypen einsetzen oder verschlüsselte Verbindungen vor dem Mounten der Dateisysteme aufbauen. Das Debugging wird erleichtert, da die Initialiserungsumgebung viele Möglichkeiten zur Fehlermeldung und -analyste bereits enthält.

Die offene Architektur des zugrundegelegten Betriebssystems bietet eine Reihe von Schnittstellen zu anderen Plattformen und ist somit nicht genuin an bestimmte Anwendungen gekoppelt. Für die gesamte Aufgabenstellung kommen etablierte Standardprotokolle und Applikationen zur Anwendung, die in den meisten Betriebsumgebungen bereits zur Verfügung

stehen. Es wird auf wohldokumentierte Progammiersprachen wie Bash und Perl zurückgegriffen, die nicht nur für das hier vorgestellte Linuxbetriebssystem zur bereitstehen.

So läßt sich mittels des Citrix-Metaframe-Clients eine Verbindung zu Windows-Servern aufbauen, die sich nahtlos in den vorhandenen Desktop eingliedern läßt. Es gibt mehrere Java-Runtime-Umgebungen, die unter Linux einsetzbar sind und damit eine ganze Reihe plattformübergreifender Software, wie z.B. Frontends zum SAP-R/3-System, unterstützen. Im Extremfall tritt vielleicht der "eigene" Desktop der vorgestellten Architektur komplett in den Hintergrund und man benutzt die Thin-Clients als preiswerte, lizenzkostenfreie, einfach skalierbare Lösung zur Anbindung an andere kommerzielle Plattformen wie Apples MAC-OS X, bzw. auch Kiosk- oder Point-of-Sale-Produkte.

A VMware auf Diskless Clients

A.1 Überblick

Im folgenden Abschnitt soll zum Thema "Einsatz der virtuellen PC-Sotware VMware" etwas weiter ausgeholt werden. Da diese spezielle Software, sie emuliert einen kompletten Intel-PC, für einige interessante Konstellationen im Zusammenhang mit Diskless Clients sorgt, erfolgt eine ausführlichere Einführung in die Funktionsweise, ihre Hardware-Anforderungen und daraus resultierenden Konfigurationsschritte. Exemplarisch beschrieben wird die Einrichtung eines Windows98 Gastbetriebssystems auf einer nichtpersistenten virtuellen Festplatte, die über NFS oder andere Netzwerkdateisysteme zur Verfügung gestellt wird.

VMware (Workstation) ist eine Applikation unter einem Betriebssystem, in der sich wiederum andere Betriebssysteme innerhalb eines Fensters der grafischen Oberfäche des Hostbetriebssystems betreiben lassen. Dazu emuliert VMware einen virtuellen PC, in dem das Gastbetriebssystem läuft. VM bedeutet Virtuelle Maschine, da nicht nur Betriebssystem-APIs, sondern ein kompletter Rechner inklusive Hardware[20] und BIOS nachgebildet werden. Das Hostbetriebssystem bezeichnet das real auf dem Rechner installierte OS, welches ein Linux oder Windows98/NT/2000/XP sein kann. Als Hardwareplattform kommt ausschliesslich die PC-Architektur in Frage. Mit dem Gastbetriebssystem ist die Installation innerhalb der virtuellen Maschine des VMware gemeint. Die Auswahl der als Gäste installierbaren Betriebssysteme ist umfangreich und umfasst Linux und FreeBSD für PC, NetWare6.0, Windows und MS-DOS in fast allen Ausführungen. Weitere OS, wie Solaris für Intel-CPU, sind denkbar, jedoch existiert keine Zusicherung seitens des Softwareanbieters.

Die VMWare Software ist eine dünne Softwareschicht, die zwischen der PC-Architektur und dem (Gast-)Betriebsystem eingeschoben wird. Sie virtualisiert die Hardware und verwaltet alle Ressourcen zur Verfügung gestellten Ressourcen. Die Grundidee stammt von der IBM-S370/390 Plattform[21]: Die vorhandene, reale Hardware wird auf eine oder mehrere virtuelle Maschinen durch virtuelle Vervielfältigung abgebildet. Eine dieser Software-Maschinen entspricht dem Hostbetriebssystem. Diese Maschine ist gegenüber den anderen herausgehoben und und koordiniert das Multiplexing. Der sogenannte VM-Monitor wird von dieser Maschine gesteuert. Er sorgt dafür, dass die verschiedenen virtuellen Maschinen die reale Hardware nicht gleichzeitig exklusiv belegen. Der VM-Monitor übernimmt in Zusammenarbeit mit dem Hostbetriebssystem die Vergabe der Ressourcen der realen Hardware. Hierbei geht es besonders darum, die sogenannten privilegierten Kommandos der Gastbetriebssysteme abzufangen, mit denen diese ihrerseits versuchen, die Ressourcen zu managen.

[20] Die Hardware umfasst neben den Basiskomponenten CPU und Speicher, IDE- und SCSI-Festplatten, CD-ROM, USB, Sound, serielle, parallele und Ethernetschnittstellen, PS2-Maus, ...

[21] dort VM/ESA

Entscheidend ist, dass die eingesetzten Gastsysteme die gleiche Prozessorarchitektur voraussetzen. So können normale Maschinenbefehle direkt von der Host-CPU ausgeführt werden. Dadurch wird das Konzept von VMware recht effizient.

Weitergehende Ansätze liefern Connectix Virtual-PC oder der freie PC-Emulator Bochs. Der Virtual-PC erlaubt auf PowerPC-Architekturen unter einem Apple-Betriebssystem einen Intel-PC zu emulieren und in diesem PC-Software ablaufen zu lassen. Der Aufwand liegt gegenpüber der VMware-Lösung höher, da nun die komplette CPU in Software nachgebildet werden muss. Bochs kann auf vielfältigen Architekturen übersetzt werden und bietet jeweils eine virtuelle PC-Hardware an, die sogar in der Lage ist bestimmte Prozessorbesonderheiten abzubilden. Dieses Projekt befindet sich jedoch noch in einem Entwicklungsstadium. Die Performance des virtuellen Rechners liegt entsprechend weit unter der der Gastarchitektur. Beide Ansätze sind für das hier beschriebene Projekt nicht verwendbar und werden nur der Vollständigkeit halber erwähnt.

A.2 Hardwareanforderungen

Möchte man Diskless Clients dazu nutzen, auf der lokalen Hardware in einer virtuellen Maschine ein weiteres Betriebssystem laufen zu lassen, addieren sich die Hardwareanforderungen des Host- und Gastbetriebssystems. Für ein rudimentäres Linux im Thin-Client inklusive der Grafikschnittstelle sind zwischen 48 und 64 MByte zu veranschlagen. Hinzu kommt der benötigte Speicherplatz der Ramdisk und des VMware. So erscheinen 128 MByte plus im VMware bereitgestellter Speicher als eine sinnvolle Größe. Da die Virualisierung der Hardware einigen Aufwand erfordert und bis zu ca. 30% Leistungseinbusse der reinen Rechenleistung der Host-CPU mit sich bringt, erscheint eine CPU ab 300 MHz ausreichend performant. Benötigt das Gastbetriebssystem selbst schon weit mehr Leistung und Speicherplatz, muss sich dieses auch in der Host-CPU und der realen Speicherausstattung niederschlagen.

Weitere Ressourcen, wie die grafische Schnittstelle mit Tastatur und Zeigegerät sowie eine Netzwerkkarte stehen standardmäßig auf einem grafischen Thin-Client zur Verfügung. Das Ensemble kann um USB und Sound ergänzt werden. Sämtliche Disketten-, CD-Rom- und Festplattenlaufwerke können von VMware emuliert werden und müssen nicht einer realen Hardware gegenüberstehen.

VMware stellt im "FullScreen-Modus" eine Besonderheit unter Linux dar: Es wird ein weiterer X-Server gestartet (Wenn "X:0" der standardmäßig laufende ist, kommt "X:1" hinzu, sonst halt "X:N+1"). Die Auflösung, welche VMware anfordert muss natürlich auf der Hostmaschine zur Verfügung stehen und in der *XF86Config* konfiguriert sein. Dabei ist es dann zu jeder Zeit möglich zwischen den verschiedenen X-Servern (also VMware im Vollbild und Fenstermodus) durch die entsprechenden CTRL-ALT-FN-Tastenkombinationen zu wechseln. Eine ähnliche Funktionalität liefert die "Mausbefreiung" CTRL-ALT unter VMware.

A.3 Der VMware-"Rechner"

Die VMware-Maschine bildet den Prozessor nach, der durch die Host-CPU direkt vorgegeben ist. Als BIOS kommt im virtuellen Rechner eines der Firma Phoenix zum Einsatz, das die üblichen Optionen traditioneller PCs kennt. Die veränderten Einstellungen des BIOS werden in einer Datei *nvram* im VMware-Verzeichnis gespeichert. Der Arbeitsspeicher wird vom Hostbetriebssystem angefordert und kann bis 1 GByte betragen unter der Voraussetzung, dass im Host diese Menge RAM real zur Verfügung steht. VMware bildet einen Standard-PCI-Grafikadapter mit VGA- und SVGA-Unterstützung bis zu einer maximalen Auflösung der Grafikauflösung des Hosts an. Die verschiedenen Auflösungen, die unter VMware genutzt werden sollen, müssen unter vom Host-Grafikadapter zur Verfügung gestellt werden.

Die virtuelle IDE-Schnittstelle bietet folgende Eigenschaften:

- Es wird ein Intel 82371 PCI Bus Master IDE-Controller für den primären und sekundären IDE-Controller verwendet.
- Es können bis zu vier IDE-Geräte, wie Festplatten oder CD-ROMs angeschlossen sein.
- Festplatten können dabei virtuelle oder physische Festplatten sein. Virtuelle Festplatten werden durch Containerdateien abgebildet, die im Dateisystem des Host-OS abgelegt werden.
- Virtuelle IDE-Festplatten können bei alten Hostdateisystemen nur bis zu 2 GByte gross sein. Sonst können sie bis auf 128 GByte gebracht werden.
- Ein CD-ROM-Laufwerk kann ein physisches Laufwerk oder eine ISO-Image-Datei sein.
- DVD-ROM-Laufwerke sind eingeschränkt nutzbar: Sie können nur zum Lesen von Daten und nicht zum Abspielen von Videos verwendet werden.

Erweitert wird die VM durch einen SCSI-Bus mit nachstehenden Eigenschaften:

- Emuliert wird ein BusLogic BT-958 kompatibler SCSI-Host-Adapter.
- Es können bis zu sieben SCSI-Geräte angeschlossen werden. Unterstützt werden Festplatten, CD-ROM-, DVD- und Bandlaufwerke. Weitere SCSI-Geräte, wie Scanner etc. sollten betreibbar sein.
- SCSI-Festplatten können als virtuelle oder physische Festplatten vorliegen, wobei für virtuelle Laufwerke die identischen Einschränkungen wie bei IDE existieren. Maximal sind bei SCSI bis zu 256 GByte Größe einstellbar.

Es können, wie im normalen Rechner auch, bis zu zwei Diskettenlaufwerke bis zu einer Kapazität von 2,88 MByte angeschlossen werden. Beim Laufwerk kann es sich um ein physisch vorhandenes Laufwerk oder eine Imagedatei handeln. Der Zugriff auf das Laufwerk erfolgt exklusiv, eine gleichzeitige Nutzung durch mehrere virtuelle Maschinen oder durch das Hostbetriebssystem sind nicht erlaubt. Die klassischen Peripherieschnittstellen, wie vier serielle und zwei parallele Ports sind konfigurierbar. Die Ausgaben können jeweils an ihre physikalischen Pendants oder in Dateien des Host-OS erfolgen. Serielle Schnittstellen erlauben zusätzlich Named Pipes. Existieren reale USB-Schnittstellen und werden diese vom Hostbetriebssystem unterstützt, können sie auch unter VMware sichtbar gemacht werden. Abgebildet werden sie als USB 1.1 Controller mit zwei Ports, an welche Drucker, Festplatten, Scanner, PDAs, Speicherkartenleser und Digitale Kameras angeschlossen werden können.

Eine Sound Blaster 16 kompatible Soundkarte bietet VMware zur Ausgabe in PCM (Pulse Code Modulation) an. Damit wird die Fähigkeit erreicht gängige Audio-Formate zu Gehör zu bringen und die Soundausgabe von Spielen zu ermöglichen. Die Qualität dieser Schnittstelle ist jedoch begrenzt und setzt selbstredend eine installierte Soundkarte mit Unterstützung seitens Hostbetriebssystems voraus. Die Zusammenarbeit mit den verschiedenen Linux-Soundtreibern klappt nicht gleich gut. Einge Soundtreiber funktionieren sowohl in der einfachen Kernel- als auch ALSA-Ausgabe, andere wiederum nicht oder sehr schlecht[22].

Ein wichtiges Feature der virtuellen Maschine bilden die bis zu drei virtuellen Ethernetkarten, der AMD PCNET Family. Mit dieser Hardware gelingt es sogar von Diskette die virtuelle Maschine diskless mit Etherboot zu starten. Zusätzlich sind bis zu neun virtuelle

[22] Zeitversatz, Stockungen, Echos, ...

Ethernet Switches enthalten, die durch vmnetN abgebildet werden. Die Ethernet-Interfaces können im Bridged, Host Only oder NAT Modus betrieben werden. Die ersten drei Stellen der MAC-Adresse sind durch VMware festgelegt und beginnen mit 00:50:56, die weiteren Stellen können beliebig gewählt werden. Die komplette MAC-Adresse jedes Adapters ist voreinstellbar.

Mit der beschriebenen Hardware-Abbildung erreicht man eine wirkliche Virtualisierung von Maschinen über die Grenzen von Hostmaschinen und -betriebssystemen hinweg. Der Satz virtueller Geräte unterscheidet sich von dem Satz der echten Hardware-Geräte, und ist, mit einigen Ausnahmen, unabhängig von der darunterliegenden Hardware.

Ausnahmen, wie der reale Prozessor, der Arbeitsspeicher und die Verfügbarkeit von USB oder Sound, beeinhalten selten eine Einschränkung bei der Portabilität von Betriebssystem-Images zwischen beliebigen virtuellen Maschinen. So gewährleistet VMware eine stabile Rechnerplattform, unabhängig von der Konfiguration der realen Maschine. Eine besondere Situation entsteht, wenn ein real installiertes Betriebssystem von einer realen Festplatte(npartition) nocheinmal in VMware gestartet werden soll. Dann unterscheidet sich zwangsläufig die beschriebene virtuelle Hardware von der real installierten. Diese Situation ist jedoch für Diskless Clients[23] irrelevant, da eine möglicherweise installierte Festplatte exklusiv VM-Gästen zur Verfügung gestellt werden kann.

Abgerundet wird die Hardware durch spezielle VMware-Softwaretreiber. Die sogenannten VMware Tools installieren spezielle Grafiktreiber, die Auflösungen über den 16 Farbmodus bei 640x480 Bildpunkten hinaus gestatten. Die Verwendung der Maus wird erleichert und einige zusätzliche Hilfen installiert. Die Treiber sind natürlich spezifisch für das Gastbetriebssystem.

A.4 Virtuelle Festplatten

Für die Benutzung von VMware mit Diskless Clients kommt in erster Linie die Virtual Disk in Frage. Der Typ Raw Disk kann eingesetzt werden, wenn ein reales Festplattenlaufwerk in der Maschine installiert und unter dem Hostbetriebssystem verwendbar gemacht wurde. Die Raw Disk kann direkt auf eine lokale IDE- oder SCSI-Festplatte oder deren Partition zugreifen. Damit kann man aus VMware heraus von einer schon existierenden Festplatten-Partition Betriebssysteme booten.

Die Virtual Disk wird durch eine Datei im Hostfilesystem abgebildet. Das Gastbetriebssystem sieht diese Festplatte als klassische physische Einheit. Dabei spielt es keine Rolle, ob sich das Dateisystem auf der Hostmaschine oder auf einem anderen Rechner im Netzwerk befindet. Die virtuelle Festplatte wird aus Sicht des Gastsystems eingerichtet und formatiert wie gewohnt. Der Vorteil liegt darin, dass keine wirkliche Partitionierung erfolgt und ein evtl. Booten der Hostmaschine entfällt. Ein weiterer erwünschter Zustand ergibt sich damit, dass die virtuelle Festplatte mit einer Minimalgröße beginnt und mit dem Bedarf bis zur eingestellten Maximalgröße wächst. Das Schrumpfen erfolgt nicht automatisch, kann aber durch VMware angestoßen werden. Eventuell muss hierfür jedoch das Gastdateisystem erst defragmentiert werden.

VMware erlaubt den Zugriff auf seine Festplatten in unterschiedlichen Modi. Im Modus "persistent" verhalten sich Festplatten wie gewohnt, alle Schreib- und Löschzugriffe werden direkt auf dem Medium ausgeführt und können nur mit den Mitteln des Betriebssystems rückgängig gemacht werden. Nonpersistente Festplatten verhalten sich wie Read-Only-Medien aus Benutzersicht. Das Gastbetriebssystem bekommt davon jedoch nichts mit, da alle Schreibzugriffe in eine spezielle Cache-Datei umgeleitet werden. Bei Lesezugriffen

[23] Diskless Client mit Festplatte erscheint als Wiederspruch in sich, gemeint ist jedoch der Betriebsmodus des Gerätes und nicht seine reale Ausstattung.

überprüft VMware auf eventuell erfolgte Schreibzugriffe und beantwortet solche Anfragen nicht vom Medium, sondern aus dem Cache. Diese Datei wird als Redo-Protokoll bezeichnet und beliebig auf einem schreibbaren Bereich des Hostdateisystems abgelegt werden. Auf festplattenlosen Maschinen muss der Ort geeignet festgelegt werden, da die Datei die Größe der zur Verfügung stehenden Ramdisk überschreiten kann. Das Redo-Protokoll wird automatisch beim Beenden einer VMware-Sitzung gelöscht.

Eine etwas abgewandelte Form liefern "Undoable Disks", welche sich ersteinmal wie nonpersistente Medien verhalten. Im Gegensatz zu einer gleichnamigen (virtuellen) Festplatte erlaubt die Undoable Disk jedoch später, Veränderungen, die im Redo-Protokoll gespeichert sind, permanent auf die (virtuelle) Festplatte zu übernehmen. Die Übernahme muss nicht am Ende einer Sitzung erfolgen, sondern kann auf spätere Sitzungen verschoben werden, wenn die Protokoll-Datei aufgehoben wird.

A.5 Die Konfigurationsdateien

Für jeden Benutzer wird in seinem Home automatisch ein Unterverzeichnis *.vmware* angelegt. Unterhalb dieses Verzeichnisses finden sich alle relevanten Dateien zur Steuerung der virtuellen Maschine(n). Die Konfigurationsdatei *.vmware/preferences* bestimmt das Verhalten des grafischen Benutzerinterfaces von VMware. Hier wird besonders das Auftauchen von Fehlermeldungen eingeschaltet oder unterdrückt:

```
01 hint.xkeymap.notLocal = "FALSE"
02 hint.mks.notLocal = "FALSE"
03 hint.usbgLinux.altuhci = "FALSE"
04 hint.gui.reset = "FALSE"
05 prefvmx.mru.suspended = ""
06 pref.toolbarIcons = "FALSE"
07 hint.disklib.lockerror = "FALSE"
08 hint.gui.poweroff = "FALSE"
09 hint.nfsmounted.persistent = "FALSE"
10 hint.nologging = "FALSE"
11 pref.motionUngrabBarrier = "1"
12 pref.motionScrollBarrier = "100000"
13 prefvmx.mru.config = "~/.vmware/autovm.conf:"
14 hint.mks.fullscreen = "FALSE"
15 pref.autoRaise = "FALSE"
16 pref.motionGrab = "TRUE"
17 pref.exchangeSelections = "TRUE"
18 pref.syncTime = "FALSE"
19 hint.cpuid.unknownfeature = "FALSE"
```

In Zeile 18 wird eingestellt, ob die VMware-interne Zeit mit der Uhr des Hostbetriebssystems synchronisiert werden soll. Zeile 19 sorgt dafür, dass Meldungen von VMware unterdrückt werden, wenn bestimmte CPU-Features nicht interpretiert werden können, z.B. das Hyperthreading beim 3 GHz Pentium 4.

Bei der Einrichtung von VMware ist es sicherlich sinnvoll sich über eine Reihe von möglichen Problemen informieren zu lassen oder Hinweise zur Laufzeit des Programmes zu bekommen. Im Einsatz für den Endbenutzer, z.B. in Kursraumumgebungen oder als Erweiterung des Linuxdesktop für nur unter Windows verfügbare Anwendungen, stören oder verwirren diese Meldungen meistens nur.

A.6 Zentrale (Hardware-)Konfiguration

VMware wird mit der Datei *vmware.conf*, wobei der Dateiname jedoch vom Benutzer auch frei gewählt werden kann, für die Benutzung mit einem bestimmten Gastbetriebssystem in einer bestimmten festgelegten Hardwarekonfiguration eingestellt. Die zuletzt geöffneten Konfigurationsdateien merkt sich VMware in den *preferences* unter "prefvmx.mru.config". Alle Dateien werden mit Doppelpunkt voneinander getrennt in einem String angegeben.

Wie auch die *preferences* handelt es sich um eine reine Textdatei, die sich ausserhalb von VMware mit einem Editor bearbeiten oder durch Skripten oder Programme dynamisch generieren läßt. Dieses geschieht im vorgestellten Projekt mittels des später erläuterten *runvmware* Shell-Skripts. In der Datei finden alle hardwarespezifischen Einstellungen statt. Die Reihenfolge der Einträge ist nicht vorgeschrieben, jedoch erhält man einen besseren Überblick bei geeigneter Gruppierung der Einträge. Zeile 1 und 2 sind VMware-spezifische Angaben, die für die Version 3.2 der Software zutreffen. Viele Einstellungen werden über Schalter realisiert, die die boolschen Werte "TRUE" und "FALSE" annehmen können.

Die Option "displayName" erlaubt das Festlegen eines Namen, der im oben Balken eines X11-Fensters nach dem vorgegebenen String "VMware Workstation:" erscheint. So lassen sich mehrere gleichzeitig laufende Instanzen unterscheiden. Die Zeilen 4 bis 6 betreffen das VMware-Fenster: Hier wird eingestellt, dass beim Start von VMware sofort die virtuelle Maschine gebootet wird. Weiterhin wird automatisch in den Full-Screen-Mode geschaltet[24] und ein Ändern der Bildschirmauflösung der virtuellen Maschine erlaubt. Beim Beenden des Gastbetriebssystems wird VMware automatisch durch die Einstellung von "gui.exitAtPowerOff" beendet.

```
01 config.version = "6"
02 virtualHW.version = "2"
03 displayName = "Test"
04 gui.powerOnAtStartUp = "TRUE"
05 gui.fullScreenAtPowerOn = "TRUE"
06 gui.fullScreenResize = "TRUE"
07 gui.exitAtPowerOff = "TRUE"
08 suspendToDisk = "TRUE"
09 apmSuspend = "FALSE"
10 hard-disk.enableIBR = "FALSE"
11 resume.repeatable = "FALSE"
12 disable_acceleration = "FALSE"
13 guestOS = "win98"
14 ide0:0.mode = "nonpersistent"
15 ide0:0.present = "TRUE"
16 ide0:0.fileName = "~/.vmware/w98"
17 ide0:1.present = "TRUE"
18 ide0:1.deviceType = "atapi-cdrom"
19 ide0:1.fileName = "/dev/cdrom"
20 floppy0.startConnected = "FALSE"
21 floppy1.startConnected = "TRUE"
22 floppy1.present = "TRUE"
23 floppy0.fileType = "device"
24 floppy1.fileType = "file"
25 floppy1.fileName = "FLOPPY-B"
```

[24] Dieser muss vom Xserver unterstützt werden und die entsprechende Auflösung verfügbar sein.

```
26 redoLogDir = ""
27 logging = "FALSE"
28 debug = "FALSE"
29 memsize = "256"
30 ethernet0.present = "TRUE"
31 ethernet0.connectionType = "bridged"
32 ethernet0.address = "00:50:56:0D:00:00"
33 sound.present = "TRUE"
34 sound.device = "/dev/dsp"
35 usb.present = "TRUE"
36 usb.generic.autoconnect = "TRUE"
37 usb.generic.devfsPath = "/proc/bus/usb"
38 mouse.fileName = "/dev/mouse"
```

Die beiden folgenden Zeilen bestimmen Funktionen des Suspend: Ein Suspend wird nicht durch APMEvents[25] eingeleitet, ein Suspend auf Festplatte jedoch möglich. Die Zeilen 10 bis 12 enthalten einige festplattenspezifische Optionen. In der darauffolgenden Zeile wird das Gastbetriebssystem spezifiziert. Ein korrekter Eintrag ist mindestens für die Installation der VMware Tools unerlässlich. Möglich sind die Einstellungen: *dos* (MS-DOS im Konfigurationseditor), *win31* (Windows3.1), *win95* (Windows95), *win98* (Windows98), *winMe* (WindowsME), *winNT* (WindowsNT), *win2000Pro* (Windows2000 Professional), *win2000Serv* (Windows2000 Server), *win2000AdvServ* (Windows2000 Advanced Server), *winXPHome* (WindowsXP Home Edition), *winXPPro* (WindowsXP Professional), als experimentell gekennzeichnete .NET-Server, *linux* (Linux), *freeBSD* (FreeBSD), *netware6* (Netware 6.0 experimentell) und *other* für nicht aufgeführte Betriebssysteme. Für diese gibt es keine speziellen VMware Tools. Alle genannten Einstellungen lassen sich über die "Misc"-Section des Konfigurationseditors erreichen. Hinzu kommt noch die Möglichkeit Debugging (Zeile 28) einzuschalten, die Lage des Red-Log zu spezifizieren (Zeile 26) und ein wiederholbares Resume auf nonpersistente Platten einzuschalten.

In Zeile 29 wird der dem Gastbetriebssystem zur Verfügung gestellte Arbeitsspeicher eingetragen, dabei berechnet VMware den für das Hostsystem benötigten Speicher ab und schlägt in Abhängigkeit des eingestellen Gast-OS untere und obere Speicherlimits vor. Die Erhöhung der Speicherzuteilung kann nur in 4 MByte Schritten erfolgen, dieses muss auch ein Skript beachten, welches den Wert aus dem real zur Verfügung stehenden Speicher ausrechnet.

Die Soundausgabe wird in den Zeilen 33 und 34 konfiguriert: Das Audiodevice ist üblicherweise */dev/dsp*. Die Verfügbarkeit wird in der nachfolgenden Zeile markiert. Nach dem fast identischen Prinzip erfolgt die Konfiguration des USB in den Zeilen 35 bis 37. Die möglichen Einstellungen sind selbsterklärend. Noch einfacher kann die Maus normalerweise mit Autodetect konfiguriert werden. Soll ein alternatives Mousedevice verwendet werden, liefert Zeile 38 das entsprechende Beispiel. Einstellungen für parallele und serielle Schnittstellen sind hier nicht gezeigt, lassen sich jedoch analog durch den Konfigurationseditor erzeugen und später auf eventuelles Skripting übertragen.

Die Zeilen 31 bis 33 demonstrieren für das erste virtuelle Ethernet-Interface die möglichen Einstellungen. Das Gerät kann ein und ausgeschaltet werden und kennt die Modi *bridged*, *hostOnly* oder *nat*. Die erste Option ist für den Betrieb im Umfeld der Diskless Clients vielleicht die interessanteste, weil dann der vorhandene DHCP-Server vom Client transparent über das bestehende Netzwerk mitgenutzt werden kann. Damit sich jeder virtuelle Client

[25] Advanced Power Management, welches das Gastbetriebssystem unterstützen muss

eindeutig identifizieren läßt, sollte man ihm eine feste MAC-Adresse zuteilen, wie es in Zeile 33 beispielhaft gezeigt wird.

A.7 Laufwerkskonfiguration

Eine der entscheidenden Aufgabe von VMware ist es, eine geeignete Zuordnung von realen und virtuellen Disketten-, Festplatten- und CD-Rom-Laufwerken zu treffen. Diese wird zum einen innerhalb der Hardwarekonfigurationsdatei eingestellt und muss zum anderen passend im virtuellen BIOS nachvollzogen werden, damit die Laufwerke für das Gastbetriebssystem auch wirklich sichtbar sind.

Für Festplatten stehen die Modi *nonpersistent*, wie im Beispiel in Zeile 14 gezeigt, *persistent* und *undoable* zur Verfügung. Diese Modi existieren für IDE- und SCSI-Festplatten. Die Nummerierung der IDE- bzw. SCSI-Busse und Anschlüsse erfolgt durch Zahlen, welche mit Doppelpunkt voneinander getrennt werden: *Bus-Nummer:Device-Nummer*. Die Einträge in den Zeilen 14 bis 16 bezeichnet das Master-Gerät, in diesem Falle eine Festplatte, am ersten IDE-Bus. Die Zeilen 17 bis 19 ein CD-Rom, welches als Primary-Slave angeschlossen ist. Die SCSI-Devices liegen auf dem Bus *0* und können von *0 - 6* durchnummeriert werden. Je nach Art einer eingestellten Festplatte verweisen die Einträge **.fileName* auf Dateien (im Beispiel Zeile 16) oder Devices (Zeile 19).

Diskettenlaufwerke werden etwas anders angesprochen und kennen die Modi *Device* (im Beispiel Zeile 23) oder *File* (Zeile 24). Dateien müssen in ihrer Größe mit der jeweiligen Nettokapazität des eingestellten Diskettenlaufwerkes im BIOS korrespondieren. Auf Devices wird im "raw"-Modus zugegriffen, wobei VMware das Gerät exklusiv nutzt. Es ist jedoch möglich über das VMware-Menu virtuelle und reale Disketten zu verbinden oder trennen, so dass ein Diskettenwechsel und eine Nutzung des Gerätes durch das Hostbetriebssystem möglich sind ohne die VM-Session zu beenden.

A.8 VMware unter Linux

Für den Einsatz von VMware werden mindestens zwei Kernelmodule benötigt, welche für die jeweils gültige Kernelversion zu übersetzen sind. Die Kernelmodule müssen passend zur eingesetzten VMware-Version gewählt werden, da eine korrekte Funktion sonst nicht gewährleistet ist. Das Modul *vmmon.o* übernimmt die Steuerung der virtuellen Maschine und die Koordination mit dem Hostbetriebssystem, wie eingangs beschrieben. Ein weiteres Modul, *vmnet.o*, sorgt für die Netzwerkschnittstellen, die für das Gastbetriebssytem zur Verfügung gestellt werden. Für die Host-Only-Anbindung wird ein eigenes Interface *vmnetN* als Gegenstück zum Netzwerkadapter innerhalb der virtuellen Maschine konfiguriert:

```
vmnet8    Link encap:Ethernet  HWaddr 00:50:56:C0:00:08
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fec0:8/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2429 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Mit einem Host-Only-Network läßt sich das Gastbetriebssytem komplett von der Netzwerkaußenwelt abschirmen. Möchte man jedoch, dass das Gast-OS nach für andere Maschinen im Netzwerk so auftritt, als wäre es ein eigener Rechner, benutzt man "Bridged-Networking". Dann können z.B. ein zentraler DHCP-Server oder externe Fileserver problem-

los mitbenutzt werden. Für das Bridged-Networking wird kein eigenes Linux-Interface angelegt, da die Ethernet-Pakete transparent an die Ethernet-Schicht des Linuxkernels durchgereicht werden.

Das Laden der VMware-spezifischen Kernelmodule geschieht mittels eigenem Runlevel-Skript (*/etc/init.d/vmware*). Damit dieses korrekt ausgeführt werden kann, muss vorher eine globale Konfiguration mittels **vmware-config.pl** erstellt werden, die auch das Verzeichnis */etc/vmware* anlegt und es entsprechend bestückt. Normalerweise generiert dieses Skript die Kernelmodule aus sich heraus, für den speziellen Diskless-Kernel kann es evtl. notwendig sein, sie per Hand zu kompilieren und in das Modulverzeichnis zu legen. **vmware-config.pl** bestimmt das Verhalten der Netzwerkschnittstellen und konfiguriert ein eigenes privates Subnetz. Dieses Subnetz und die dafür von VMware angebotenen DHCP- bzw. Samba-Server sind für einen Einsatz auf festplattenlosen Maschinen weniger interessant.

Das Runlevel-Skript wird für die DXS-Umgebung etwas erweitert, um sicherzustellen, dass spezifische Module, wie *floppy.o* für die realen und *loop.o* für die virtuellen Disketenlaufwerke geladen sind. Nach dem Laden der Module wird eine Disketten-Image-Datei loopback eingemountet, damit sie später auch nichtprivilegierten Benutzern zur Verfügung steht.

A.9 Einsatzbereiche

VMware kann in einer ganzen Reihe Szenarien benutzt werden, welche keine exorbitanten Anforderungen an die Grafik-, Festplatten- und Speicher-IO-Leistung stellen. Generell kann man davon ausgehen, dass die Leistungseinbußen der reinen CPU-Power zwischen 10-30 % liegen. Die Festplattenperformance kann nicht mit der realen Hardware mithalten und nicht alle Geräte, wie z.B. DVD oder Sound können exakt abgebildet werden. Für festplattenlose Umgebungen kommen in erster Linie virtuelle Disks in Frage, die (non)persistent oder rückgängig-machbar betrieben werden. *Nonpersistente* Festplatten sind für eine Reihe der nachfolgend beschriebenen Szenarien geeignet. Man kann auf diese Weise die virtuelle Maschine immer auf ein definiertes Niveau bringen und im gleichen Status starten. Beispiele dafür, wofür man *undoable* Festplatten einsetzen kann, sind unter anderem die Installation von Software oder das Ausführen von Verwaltungsaufgaben, die eventuell später, wenn Probleme auftreten, rückgängig gemacht werden müssen. Folgende Szenarien sind vorstellbar, wovon einige sich aus der besonderen Konstellation der DXS heraus ergeben:

- Kursarbeitsplätze und Schulungsrechner
- Ergänzung des eingesetzten Host-OS durch externe Applikationen
- Langzeitarchivierung von Betriebssystemen und speziellen Anwendungen
- Testumgebungen für Softwareentwicklung, -erprobung und Demonstration
- Ausstattung von Hotline- und Beratungsarbeitsplätzen
- "Stabilisierung" von in der Administration aufwändigen OS
- Schaffung einer "transportablen" Arbeitsumgebung

Ein wichtiges Einsatzfeld sind Schulungsräume in denen für verschiedene Kurse unterschiedliche Betriebssysteme präsentiert und vielleicht nur mit sehr kurzen Zwischenpausen genutzt werden sollen. Hier kommt es selten auf die reine Performance an, sondern mehr auf das generelle "Gefühl" im Umgang mit einem spezifischen OS. Arbeitet man mit virtuellen Festplatten kann ein einziges geeignet vorbereitetes Image zentral zur Verfügung

gestellt oder an die einzelnen Clients verteilt werden. Im besten Falle gelingt es ein einziges *nonpersistent* Image für alle Systeme gleichzeitig z.B. über NFS einzubinden. So wird sichergestellt, dass alle TeilnehmerInnen eines Kurses von den gleichen Startbedingungen ausgehen. Für diesen aus Sicht des Autors optimalen Fall sind jedoch eventuelle Spezialanpassungen des Gastbetriebssystems notwendig. Das betrifft insbesondere die Microsoft-Produkte. Ein Beispiel für Windows98 wird im folgenden Abschnitt beispielhaft erläutert. Eine besondere Kurssituation bietet sicherlich das "nicht-kooperative" Benutzerumfeld in Schulen: Zwar sind diese inzwischen mit Rechnern weitgehend ausgestattet, jedoch bietet eine klassische Consumer-Windowsinstallation kaum die optimale Arbeitsgrundlage. Häufig sind die Maschinen verstellt, die Softwareinstallationen nicht einheitlich oder vollständig oder die Geräte so manipuliert, dass wenig funktioniert.

In einem nächsten Schritt können Kursleiter ihre Arbeitsumgebungen bei der Vorbereitung, z.B. zu Hause, in einer eigenen VMware-Installation anpassen und notwendige Software installieren. Das fertige Image wird im Kurs an alle Teilnehmer verteilt oder diesen zentral zur Verfügung gestellt. Aufwändige Koordination zwischen Technikern und Kursleitern wird unnötig, nachträgliche Anpassungen der einzelnen Arbeitsplätze kann entfallen. Neue Modelle eines Kursbetriebes werden möglich. Dadurch erhält man eine transportable Umgebung in der ein mit allen wichtigen Daten vorkonfigurierter Arbeitsplatz rechnerunabhängig benutzt werden kann. So könnte z.B. der eigene Desktop auf dem Leih-Laptop zu einem Kongress mitgenommen werden, ohne dass spezielle Anpassungen des OS an die neue Hardware notwendig wären. Der Transport solcher speziellen VM-Images stellt aufgrund keine größeren Probleme mehr dar: CD-Brenner sind weitverbreitet und die Verwendung von CD-RW Rohlingen bietet Platz für bis zu 700MByte große Images, die für viele Szenarien ausreichen. Weiterhin denkbar und bald auch bezahlbar werden USB- oder CompactFlash-Memory-Karten oder DVD-RW-Lösungen.

Stehen bestimmte Applikationen unter einem OS nicht zur Verfügung kann VMware Abhilfe schaffen. Solche Programme werden dann unter dem passenden Betriebssystem in einer VM-Box aufgerufen. Jedoch ist der Datenaustausch nicht so komfortabel wie unter einer einheitlichen Plattform oder Benutzeroberfläche. Er muss über den Umweg der Netzwerkspeicherung oder Diskettenimages geführt werden.

Ein anderes Einsatzgebiet ist die Langzeitarchivierung von Software und Dokumenten. Mit VMware können Betriebssysteme in einer bestimmten Konfiguration "eingefroren" werden ohne dass hierfür separate Hardware vorgehalten werden muss. Stellt man eine solche Umgebung wiederum auf *nonpersistent* können keine ungewollten Änderungen am Archivsystem erfolgen. Auf diese Weise kann ältere Software, für die es keine Portierungen auf aktuelle Betriebssysteme gibt, weitergenutzt werden ohne dass man auf die Vorteile aktueller Betriebssysteme verzichten muss. Diese Applikationen werden in der VM-Box ausgeführt; die geschilderten Performancenachteile gegenüber einer nativen Benutzung fallen aufgrund des Alters der Software nicht ins Gewicht.

DOS, Windows3.1 oder 95, alte Linux-Installationen können auf einer einzigen Maschine zusammengefasst werden und so Software-Beratern oder Hotlines bequemen Zugriff auf vergangene Welten verschaffen. Diese Betriebssysteme sollten allein aus Sicherheitsgründen nicht stand-alone in einem Netzwerk laufen, da wichtige Sicherheitspatches und Updates nicht verfügbar sind. Gleichzeitig war die Netzeinbindung zumindest einiger genannter Kandidaten eher rudimentär, welches sich über den Umweg von VMware in Verbindung mit dem X11- oder VNC-Protokoll aufheben lässt.

Ein ähnliches Szenario bietet die Migration von Arbeitsplätzen von einem Betriebssystem in ein anderes. Möchte man aufgrund der Lizenzkostenentwicklung von Microsoftbetriebssystemen am Arbeitsplatz wegkommen, kann ein schrittweiser Umstieg mittels Umweg über VMware geführt werden. Das vormals installierte OS wird weiterhin auf der Maschi-

ne ausgeführt, nun jedoch als Gastbetriebssystem. Der Anpassungsaufwand wird nicht in die Migration von z.B. Windows98 auf WindowsXP investiert, sondern fließt in die Realisierung eines Linux-Desktop. Wesentliche Kosteneinsparungen ergeben sich auf die Lizenz des Betriebssystems nicht sofort, können jedoch bezogen auf die Applikationen und spätere Updates erheblich werden.

Betriebssysteme, welche einen sehr hohen Betreuungsaufwand erfordern, wie Windows95, -98, -ME, lassen sich durch VMware absichern. Bootet der Benutzer, der auf diesen Consumer-OS automatisch Systemadministrator ist, von einem *nonpersistent* Image, sind fatale Modifikationen nur für eine Sitzung gültig. Dieses betrifft das versehentliche Löschen wichtiger Systemdateien, die "Installationen" von Viren und Backdoors oder die Performance ruinierende oder das Netzwerk belastende Gimmicks wie aufwändige Bildschirmschoner.

Weitere Beispiele sind konstante Umgebungen für Software-Tester und technische Unterstützung als auch Software-Demonstrationen. Wenn eine Softwareentwicklung auf mehreren Plattformen erfolgt, müssen diese nun nicht getrennt vorgehalten werden, fatale Abstürze wirken sich nicht negativ auf *nonpersistente* Betriebssysteme aus. Ähnliches gilt für Webentwicklunger, die verschiedene Browser in unterschiedlichen Umgebungen testen wollen.

Die DXS-Installation bringt eine weitere Anwendung für eine VM-Box mit sich: Für eine bequeme Einrichtung der Software für die Clients bei getrennten Dateisystemen bietet sich die Benutzung von VMware an. So kann die Installation und das Software-Update weiterhin bequem auf dem Server erfolgen ohne dafür eine eigene Hardware für einen Speziel-Client aufzustellen.

A.10 Windows98 im DXS-Umfeld

Windows98 (zumindest in der sogenannten Second Edition) bildet den Quasi-Standard unter den Windows-Betriebssystemen mit Consumer-Ausrichtung. Fast die gesamte verfügbaren Anwendungssoftware für Windows ist auf dieser Plattform lauffähig. Legt man nicht besonderen Wert auf bestimmte Features von WindowsXP, der Zukunft der MS-Betriebssysteme, kann man mit geeigneten Einschränkungen mit Windows98 noch sehr gut arbeiten. Aus einem Betriebssytem, welches als Admins-Horror unter den Gesichtspunkten einer Stand-Alone-Installation zu bezeichnen ist, wird ein benutzbarer Desktop. Dieses gelingt, indem Windows98 innerhalb VMware auf Basis von Diskless X-Stations betrieben wird. Optimalerweise verwenden alle Clients mit einem recht ähnlichen Anwendungsprofil, wie Kursräume oder Verwaltungen, ein einziges gemeinsames Image, welches über einen leistungsfähigen NFS-Server bereitgestellt wird.

Damit sich die Clients dabei nicht ins Gehege kommen, sind einige Schritte notwendig: Zum einen darf kein Client das gemeinsame Image verändern dürfen, also wird es nonpersistent in einem ReadOnly-Bereich des Dateisystems bereitgestellt. Zum anderen müssen sich die Client-Images in einigen zentralen Parametern zwingend Unterscheiden: Der Windows-Name der Maschine und die IP-Konfiguration zählen hierzu. Dieses geschieht entweder über das Laden von Registry-Einträgen beim Bootvorgang oder dynamische IP-Zuweisung per DHCP. Damit keine Locking-Probleme für das gemeinsam genutzte Image entstehen, legt jeder Client für sich einen Link auf das Image an. Im gemeinsamen Verzeichnis mit diesem Link wird das Lockfile generiert. Soll die Datei mit den virtuellen Änderungen am Windows98-Festplatten-Image nicht im */tmp*[26] landen, ist ein entsprechender Eintrag in der VMware-Konfiguration unumgänglich.

Das Einlesen der Registry-Einträge muss zu einem frühen Zeitpunkt geschehen und wird

[26] Hier können Probleme mit dem Speicherplatz auftreten, wenn die Datei direkt in die Ramdisk geschrieben wird. In Abhängigkeit der Art der Windows98/VMware-Sitzung kann die Redo-Datei den Umfang etlicher Megabyte bekommen.

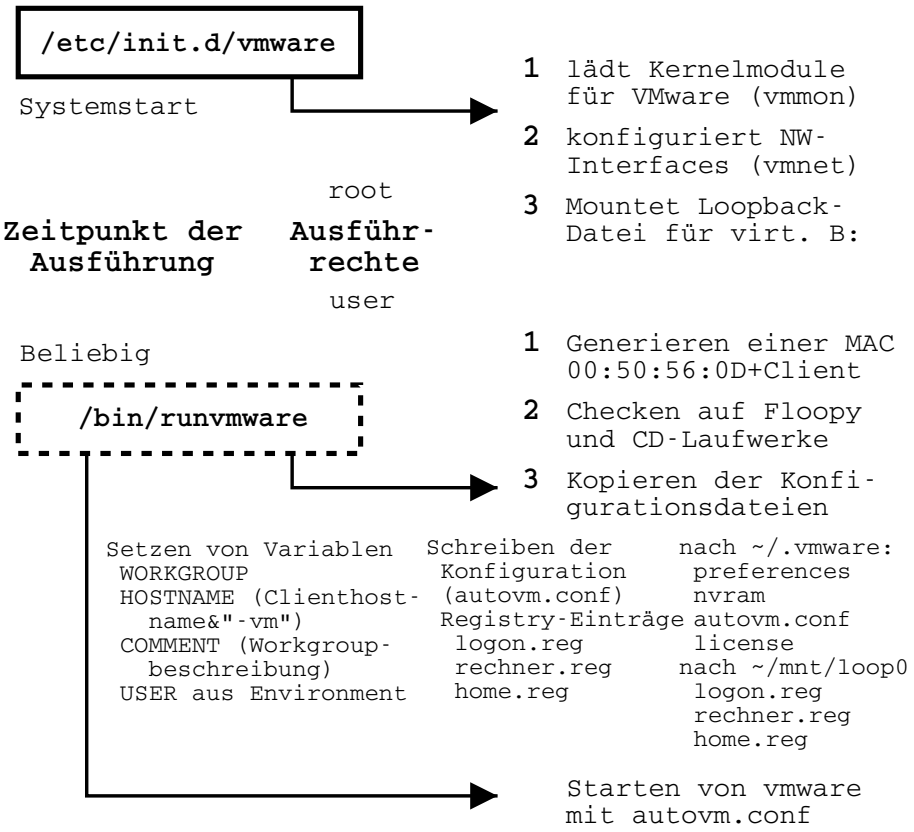


Abbildung 4: Ablauf der Einrichtung von VMware

deshalb aus der *autoexec.bat* über den Umweg der *run.bat* angestoßen. Auf diese Weise muss im gemeinsam genutzten Festplattenimage eine einzige Zeile in die *autoexec.bat* eingetragen werden, alles evtl. zwischen den verschiedenen virtuellen Hosts Variable geschieht über die *run.bat*. Es gibt drei Gruppen von Registry-Einträgen, die aus letzterer heraus erfolgen:

- *home.reg* sorgt dafür, dass das Homeverzeichnis des sich anmeldenden Benutzers automatisch (nach Eingabe seines Passworts bei der Anmeldung) eingebunden wird. Damit steht dem Benutzer nichtflüchtiger Speicherplatz im virtuellen Windows98-Rechner über ein Netzlaufwerk zur Verfügung. "SMBSERVER", "USER" und der Laufwerksbuchstabe können mittels dieser Datei festgelegt werden:

```
REGEDIT4

[HKEY_CURRENT_USER\Network\Persistent\H]
"RemotePath"="\\\\SMBSERVER\\USER"
"UserName"="USER"
"ProviderName"="Microsoft Network"
```

- *logon.reg* trägt den Benutzer in die Windows-Anmeldemaske ein, der gerade die virtuelle Maschine ausführt. Diese Information wird der *USER*-Variablen entnommen.

```
REGEDIT4

[HKEY_LOCAL_MACHINE\Network\Logon]
```

```
"username"="USER"
"PrimaryProvider"="Microsoft Network"
"LMLogon"=hex:00,00,00,00
```

- *rechner.reg* konfiguriert das "Windows-Netzwerk": Festgelegt werden kann der Windows-Name der virtuellen Maschine, der Name der Arbeitsgruppe und ein Kommentar.

```
REGEDIT4

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"ComputerName"="CN"
"Workgroup"="WG"
"Comment"="CMT"
"StaticVxD"="vnetsup.vxd"
"Start"=hex:00
"NetClean"=hex:01

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\
                                Control\ComputerName\ComputerName]
"ComputerName"="CN"
```

Leider entfällt diese einfache und elegante Möglichkeit für die "professionellen" Windowsversionen der NT- und XP-Linie. Hier sind einige Anstrengungen notwendig, um die Registry rechtzeitig mit den benötigten Einträgen zu befüllen.

Die IP-Konfiguration geschieht am besten mittels DHCP, das bereits zur Einrichtung der Diskless Clients eingesetzt wird. Um jede virtuelle Maschine eindeutig benennen zu können, sorgt das *runvmware*-Skript dafür, dass jede Maschine einige eindeutige MAC-Adresse erhält, die sich aus vier Stellen für VMware und den zwei letzten Stellen der realen MAC-Adresse des Clients zusammensetzt. Zur Abgrenzung von der realen Hardware bekommt der Windowsname an den Client-Hostnamen noch ein "-vm" angehängt.

Alle persönlichen Einstellungen müssen die Benutzer dieser Lösung in ihrem Netzlaufwerk oder auf Diskette vornehmen. Weitergehende Konfigurationen wären durch das Managen von Profilen auf dem Samba-Server möglich, somit entfällt auch hier der Eingriffsgrund für Administratoren.

B Skripten

B.1 Einsatz von Skripten

Die im folgenden vorgestellten Skripten erledigen eine Reihe von Aufgaben, welche den Betrieb von Linux Diskless Clients vereinfachen: In den Kreis der Konfigurationsskripten fallen **mk_dxsfs.sh** und **mkdxsinittar**, welche das Basisdateisystem der Thin-Clients erzeugen bzw. das Ramdisk-Image generieren.

Weitere Skripten werden zum Zeitpunkt des Startens der Diskless Clients ausgeführt und sorgen für die Laufzeitkonfiguration dieser Systeme. Zu aller erst wird in der Ramdisk anstelle des klassischen **init** ein dieses ersetzendes Shellskript ausgeführt, das die Hardware für den Netzbetrieb vorbereitet, die Konfiguration mittels **dhclient-script** anstößt und das Root-Verzeichnis der Maschine erstellt. Die Erstellung einer *dhcpd.conf* kann durch den Aufruf von **dhcp-generate -c** interaktiv erfolgen.

Die Konfiguration einiger Hardware-Komponenten, wie der XFree86-Grafik, die Einrichtung der Open-GL-Unterstützung, die Auswahl des Audio-Treibers und die Erstellung der

Mount-Points erfolgt mittels des Skriptes **hwsetup**. Der Betriebsmodus für die grafische Oberfläche wird mittels **startgui** ausgewählt. **runvmware** ist ein Shell-Skript welches mit normalen Userprivilegien aufgerufen werden kann, um ein vorkonfiguriertes VMware zu starten.

Die Skripten sind in der Reihenfolge ihrer Ausführung gelistet. Sie sind nicht immer in der jeweils aktuellsten Ausgabe angefügt. Diese beziehe man am besten über die Webseite aus dem Gesamtpaket.

B.2 Einrichtungsskripten

Das Root-Filesystem der DXS wird durch den Aufruf von **mk_dxsfs.sh** im ausgepackten Unterverzeichnis der Diskless-Client-Umgebung gestartet. Es fragt vom Benutzer einige Parameter ab, die z.B. für die Einrichtung der */etc/exports* des Servers und der */etc(.s)/auto.home* der Clients verwendet wird. Zusätzlich kann ein Kernel inklusive gewünschter Konfiguration, wie z.B. Splash-Image oder Progress-Patch ausgewählt werden. Das Skript triggert seinerseits **mkdxsinittar**, welches die INITTAR-Umgebung erzeugt und ins Bootverzeichnis der Client-Kernel kopiert.

```
#!/bin/sh
#
# script for generating dxs filesystem from existing linux installation
#
VERSION="0.5.1e"
#
# We assume four nfs shares exported:
# 1) the root filesystem without /usr, /opt, /tmp/users which is created
#    here
# 2) /usr, /opt directly exported from server filesystem
# 3) /tmp/users as a special rw exported share for temporary files
#
# Dirk von Suchodoletz <dirk@goe.net>, 11-02-2003
#
##############################################################################

# helper functions
#
# compute the subnet address of the local network
ipv4_subnet() {
  local ip="$1"
  local netmask="$2"

  # Split quad-dotted addresses into bytes
  # There is no double quote around the back-quoted expression on purpose
  # There is no double quote around $ip and $netmask on purpose
  set 'IFS='.'; echo $ip $netmask'

  echo $(($1 & $5)).$(($2 & $6)).$(($3 & $7)).$(($4 & $8))
}
# compute the broadcast address
ipv4_broadcast() {
  local ip="$1"
  local netmask="$2"

  # Split quad-dotted addresses into bytes
  # There is no double quote around the back-quoted expression on purpose
  # There is no double quote around $ip and $netmask on purpose
  set 'IFS='.'; echo $ip $netmask'

  echo $(($1 | (255 - $5))).$(($2 | (255 - $6))).$(($3 | (255 - $7))).$(($4 | (2
55 - $8)))
}

#
# declare some variables and set some defaults
NFSROOT="/nfsroot"
```

```
NETNAME=‘route -n | grep eth0 | grep -v "UG" | awk ’{ print $1 }’‘
NETMASK=‘route -n | grep eth0 | grep -v "UG" | awk ’{ print $3 }’‘
#NETMASK=‘ifconfig eth0|grep "inet addr"|awk -F : ’{print $4 }’|awk ’{print $1}’‘
SERVER=‘ifconfig eth0|grep "inet addr"|awk -F : ’{print $2 }’|awk ’{print $1}’‘
if [ -z $NETNAME ] ; then
NET="192.168.2.0/255.255.255.0"
else
NET="$NETNAME/$NETMASK"
fi
PROGRESS=0
ETCSUBDIRS="/etc/cron.daily /etc/cups /etc/default /etc/java /etc/profile.d \
/etc/unixODBC /etc/gimp /etc/gtk /etc/pam.d /etc/skel /etc/iproute2 /etc/xml \
/etc/WindowMaker /etc/ssl /etc/security /etc/alsa.d /etc/gpm /etc/openldap \
/etc/opt /etc/postfix /etc/ssh /etc/cron.d /etc/sane.d /etc/libiodbc \
/etc/radiusclient /etc/diameter /etc/raddb /etc/ipsec.d /etc/texmf \
/etc/samba /etc/ximian /etc/stunnel /etc/hotplug /etc/vmware \
/etc/SuSEconfig"
ETCFILES="bash.bashrc profile protocols shells ttytype vimrc wgetrc \
enscript.cfg esd.conf zshenv zshrc networks netgroup pluggerrc mesa.conf \
hosts.allow hosts.deny hosts.equiv securetty codecs.conf input.conf \
mplayer.conf fam.conf lesskey lesskey.bin mime.types raw rpc screenrc"

# check files and directories for existence
TMPVAR=""
for check in $ETCSUBDIRS; do
test -d $check && TMPVAR=$TMPVAR" "$check;
done
ETCSUBDIRS=$TMPVAR
TMPVAR=""
for check in $ETCFILES; do
test -f /etc/$check && TMPVAR=$TMPVAR" "$check;
done
ETCFILES=$TMPVAR

(( NARG=0 ))
TEST=( $@ )
while [ $NARG -le $# ] ; do
case "${TEST[$NARG]}" in
*help*)
echo -n "$0 generates dxs filesystem from existing "
echo -ne "linux installation\nPass nfsroot with the"
echo -e "$0 -nfsroot <value> option\n"
exit 0
;;
*nfsroot*)
NFSROOT=${TEST[$NARG+1]}
(( NARG=$NARG+1 ))
;;
*exp*)
NET=${TEST[$NARG+1]}
(( NARG=$NARG+1 ))
;;
esac
(( NARG=$NARG+1 ))
done

echo "Welcome to version $VERSION of mk_dxsfs.sh!"
echo
echo "Please answer the following questions: (Enter takes defaults)"
echo
echo "Which network do you want to use for DXS? (A.B.C.0)"
echo -n "* [ $NETNAME ] "
read NN
if [ -z NN ] ; then NETNAME=NN; fi
echo "Which netmask should be used? (255.B.C.0)"
echo -n "* [ $NETMASK ] "
read NM
if [ -z NM ] ; then NETMASK=NN; fi
echo "Using $NETNAME/$NETMASK !"
KERNEL=3
echo "Which kernel should be used?"
```

```
echo "Dont forget to match your dhcpd.conf settings to the appropriate"
echo "framebuffer resolutions: lpp->0x301, splash->0x317!"
echo "  [1] 2.4.18 splash"
echo "  [2] 2.4.18 nosplash"
echo "* [3] 2.4.19 splash"
echo "  [4] 2.4.19 nosplash"
echo "  [5] 2.4.19 lpp"
read KRN
if [ -z KRN ] ; then KERNEL=KRN; fi
echo "Where automount home directories from? (A.B.C.D:/home-dir)"
echo -n "* [ $SERVER:/home ]"
read AM
if [ -z AM ] ; then AMT=AM;
else AMT=$SERVER:/home; fi

echo "This script sets up the basic filesystem structure for diskless"
echo "X-stations now. Please enable the NFS-Share $NFSROOT/dxs in your"
echo "servers /etc/exports file, i.e.:"
echo "$NFSROOT/dxs  $NET(ro,no_root_squash)"
echo "/tmp/dxs $NET(rw,no_root_squash)"
echo "/usr $NET(ro)"
echo "/opt $NET(ro)"
echo
echo "If you would like to use TeX too enable the following lines"
echo "/var/lib/texmf $NET(ro)"
echo "/var/cache/fonts $NET(rw)"
echo
echo "... and (re)start your nfs server."
echo

# create mountpoint for dxs tmp directory
echo -e "Creating dxs tmp directory in '/tmp/dxs'. Do not delete it! Change\
\nmountpoint in dhclient-script instead if you like to place it elsewhere!\n"
mkdir -m 1777 /tmp/dxs || { echo -e "Failed to create /tmp/dxs directory \
for diskless X stations!\n(remove old entries before.)"; exit 1; }

# checking for directory entries to create
test -f exclude || { echo "File with exclude list for DXS filesystem\
 not found!"; exit 1; }
# build an exclude list - extract the directories from 'exclude'
DIR=''; for i in `cat exclude` ; do if [ -d $i ] ; then if [ $DIR ] ;
then DIR="$DIR|(^$i)"; else DIR="(^$i)"; fi ; fi ; done
# exclude backup files too
if [ -z $DIR  ] ; then
  DIR="*.*save|*.*bak|*.*old";
  else DIR="$DIR|*.*save|*.*bak|*.*old";
fi
# exclude the files (hardlinks) from 'exclude'
HL=''; for i in `cat exclude` ; do if [ -f $i ] ; then if [ $HL ] ;
then HL="$HL|($i)"; else HL="($i)"; fi ; fi ;  done
# exclude backup files too
if [ -z $HL ] ; then
  HL="*.*save|*.*bak|*.*old|*~";
  else HL="$HL|*.*save|*.*bak|*.*old|*~";
fi
# exclude the softlinks listed in 'exclude'
SL=''; for i in `cat exclude` ; do if [ -L $i ] ; then if [ $SL ] ;
then SL="$SL|($i)"; else SL="($i)"; fi ; fi ; done
if [ -z $SL ] ; then
  SL="empty-string";
fi

# untarring basic filesystem structure
echo -n "creating basic filesystem structure: "
test -f dxs.tgz || { echo "File with basic filesystem structure for DXS \
 not found!"; exit 1; }
test -d $NFSROOT || mkdir -p $NFSROOT ||
{ echo "problems creating nfsroot -> $NFSROOT"; exit 1; }
echo -n ".."
for i in `tar -xvpzf dxs.tgz -C $NFSROOT` ; do
(( PROGRESS=$PROGRESS+1 ))
```

```
if [ $PROGRESS -ge 10 ] ; then echo -n "." ; PROGRESS=0; fi
done
echo " done"

# fake link for easier configuration (removed later)
ln -fs $NFSROOT/dxs/etc.s $NFSROOT/dxs/etc

# creating directories
echo -n "creating directories: "
for i in `find /bin /sbin /lib $ETCSUBDIRS -type d 2>/dev/null | \
grep -E -v $DIR` ; do
(( PROGRESS=$PROGRESS+1 ))
if [ $PROGRESS -ge 10 ] ; then echo -n "." ; PROGRESS=0; fi
test -d $NFSROOT/dxs$i || test -L $NFSROOT/dxs$i \
|| mkdir -p $NFSROOT/dxs$i
done
echo " done"

# hardlinking or copying files
CMD="ln"
DO="hardlinking"
ln /bin/mount $NFSROOT/dxs/bin/mount &>/dev/null || \
{ CMD="cp -a"; DO="copying"; }
echo -n "$DO files: "
for i in `find /bin /sbin /lib $ETCSUBDIRS -type f | grep -v -E "$DIR" \
| grep -E -v $HL` ; do
(( PROGRESS=$PROGRESS+1 ))
if [ $PROGRESS -ge 10 ] ; then echo -n "." ; PROGRESS=0; fi
test -f $NFSROOT/dxs$i || test -L $NFSROOT/dxs$i \
|| test -e $NFSROOT/dxs$i || $CMD $i $NFSROOT/dxs$i
done
echo " done"

# softlinking files
echo -n "adding soft links: "
for i in `find /bin /sbin /lib $ETCSUBDIRS -type l | grep -E -v $DIR\
| grep -E -v $SL` ; do
(( PROGRESS=$PROGRESS+1 ))
if [ $PROGRESS -ge 10 ] ; then echo -n "." ; PROGRESS=0; fi
test -L $NFSROOT/dxs$i || cp -a $i $NFSROOT/dxs$i
done
echo " done"

# softlinking /etc subdirs
for i in $ETCSUBDIRS ; do
i=`echo $i | sed -e s,/etc/,,i`
test -L $NFSROOT/dxs/var/ram/etc/$i || \
ln -s /etc.s/$i $NFSROOT/dxs/var/ram/etc/$i
done

# copy the chooser binary
cp /etc/X11/xdm/chooser $NFSROOT/dxs/etc.s/X11/xdm

# hardlinking some standard /etc files (add all the other to the ETCFILES
# variable you will need for your applications)
for i in $ETCFILES ; do
ln -f /etc/$i $NFSROOT/dxs/etc.s/$i
test -L $NFSROOT/dxs/var/ram/etc/$i || \
ln -s /etc.s/$i $NFSROOT/dxs/var/ram/etc/$i 2>/dev/null
done

# touch the modules.dep files
touch $NFSROOT/dxs/lib/modules/*/modules.* &

# change fake link back to original one
rm $NFSROOT/dxs/etc
ln -fs /RAM/etc $NFSROOT/dxs/etc

# configuring automounter for home directories
echo -e "# /etc/auto.home\n#\n# created by mk_dxsfs.sh $VERSION\n" \
    >$NFSROOT/dxs/etc.s/auto.home
```

```
echo -e "*\t-rsize=8192,wsize=8192,rw\t$AMT/&" \
    >>$NFSROOT/dxs/etc.s/auto.home

echo "Please check all configuration files for correctness!"

#echo "Merging boot kernel with the ramdisk file"
#[ -f $dxsdir/boot/bootimg ] && \
# cp $dxsdir/boot/bootimg $dxsdir/boot/bootimg.old
#MKNBI=‘which mknbi-linux‘
#[ x$MKNBI="x" ] && { echo "Command ’mknbi-linux’ not installed."; exit 1; }
#mknbi-linux --output=$NFSROOT/dxs/boot/bootimg \
# $NFSROOT/dxs/boot/bzImage inittar.dxs

# do some sanity checks
echo "Doing some checks on your installation ..."
grep -s "$NFSROOT/dxs.*no_root_squash" /etc/exports &>/dev/null || \
echo -e "->> Problem occured: <<-\n\
You should have an entry for the diskless client root filesystem \
\nlooking like:\n $NFSROOT/dxs 1.2.3.4/255.255.255.0(ro,no_root_squash)\n\
Do not forget the ’no_root_squash’ option! It is important for running \
\ncommands (like devfsd) with root priviliges!"
grep -s "/usr" /etc/exports &>/dev/null || \
echo -e "->> Problem occured: <<-\n\
You should permit your clients to use your servers ’/usr’!"
grep -s "/opt" /etc/exports &>/dev/null || \
echo -e "->> Problem occured: <<-\n\
You should permit your clients to use your servers ’/opt’!"
[ -x $NFSROOT/dxs/sbin/devfsd ] || \
echo -e "->> Problem occured: <<-\n\
Could not find the ’devfsd’ (device filesystem daemon) in \
$NFSROOT/dxs/sbin!\n You will need it to properly run your machines."
[ -x $NFSROOT/dxs/sbin/dhclient ] || \
echo -e "->> Problem occured: <<-\n\
Could not find the ’dhclient’ programm to get configuration data \
from\n dhcp servers! You do not will succeed booting your diskless clients."
which mknbi-linux &>/dev/null || \
echo -e "->> Problem occured: <<-\n\
Could not find the ’mknbi-linux’ programm! It is needed to tag the boot \
kernel for use with etherboot."
which gethostip &>/dev/null || \
echo -e "->> Problem occured: <<-\n\
Could not find the ’gethostip’ programm! It may be needed for pxelinux.cfg \
configuration file naming. May be ignored if etherboot is the only choice."

# generate pxelinux.cfg network configuration file (quick hack!)
PXECFG="boot/pxelinux.cfg/"‘gethostip $NETNAME|awk ’{print$3}’||echo default‘
echo -e "# pxelinux.cfg/$NETNAME file\n# for instructions see syslinux \
package by Peter Anvin\n#\n# Setup for linux diskless clients in network \
$NETNAME\n#\n# Dirk von Suchodoletz <dirk@goe.net>, 20-01-2003\n#\n \
# created by mk_dxsfs.sh $VERSION\n" > $NFSROOT/dxs/$PXECFG
echo -e "label linux\n\tkernel bzImage" >>$NFSROOT/dxs/$PXECFG

# link bzImage to the kernel file choosen
RPT=‘pwd‘
cd $NFSROOT/dxs/boot
case $KERNEL in
    1|2)
ln -sf bzImage-2.4.18 bzImage
echo -e "\tappend initrd=inittar.nosplash">>$NFSROOT/dxs/$PXECFG
;;
    3|4)
ln -sf bzImage-2.4.19 bzImage
echo -e "\tappend vga=0x317 initrd=inittar.splash debug=0">>$NFSROOT/dxs/$PXECFG
;;
    5)
   ln -sf bzImage-2.4.19-lpp bzImage
echo -e "\tappend vga=0x301 initrd=inittar.nosplash">>$NFSROOT/dxs/$PXECFG
;;
esac
cd $RPT
```

```
# setting up boot kernel dependend on SuSE version
if test -f /etc/SuSE-release -a -f /bin/splash ; then
    VER=`cat /etc/SuSE-release | grep VERSION | sed -e "s,.*= ,,g"`
case "$VER" in
        8.0|8.1)
    echo "Creating INITTAR file now (8.X versions) ..."
        echo "Running 'mkdxsinittar -dxsrd \"$NFSROOT/dxs\" -k \"bzImage\"' ..."
    ./mkdxsinittar -dxsrd "$NFSROOT/dxs" -k "bzImage"
;;
*)
    echo "Creating INITTAR file now (7.3 and older) ..."
    echo "Running 'mkdxsinittar.old -dxsrd \"$NFSROOT/dxs\" -k \"bzImage\"' ..."
    ./mkdxsinittar.old -dxsrd "$NFSROOT/dxs" -k "bzImage"
;;
esac
else
    echo "Creating INITTAR file now without splash"
echo "Running 'mkdxsinittar -dxsrd \"$NFSROOT/dxs\" -k \"bzImage\"' ..."
./mkdxsinittar -dxsrd "$NFSROOT/dxs" -k "bzImage"
fi

echo "... done"
exit 0
```

Das temporäre Dateisystem der Ramdisk wird durch **mkdxsinittar** angelegt. Es kennt die Kommandozeilenparameter "-drd" für das Root-Verzeichnis der Clients (default: */nfsroot/dxs* und "-k" für das Kernelimage (default: *bzImage*). In Abhängigkeit der Auswahl des Kernels werden die geeigneten Module ausgewählt und bei Bedarf ein Splash-Image angefügt. **mkdxsinittar** generiert sowohl für die Benutzung von Etherboot (Ergebnis-Image: *bootimg*, als auch die Verwendung von PXE (Ergebnis-Image: *inittar.splash* und *inittar.nosplash* geeignete Dateien. Wenn das Paket **busybox** zur Verfügung steht, wird dieses anstelle einer Standalone-Ash eingesetzt. Damit stehen automatisch einigezusätzliche Standard-Tools, wie **ls** im INITTAR für ein eventuell notwendiges Debugging zur Verfügung, nachdem man den entsprechenden Link auf **busybox** gesetzt hat.

```
#!/bin/bash
#
# mkdxsinittar - create the inital ramdisk images for linux diskless clients
# usage: see below usage() or call with -h
#
# Version 0.0.4d
#
# Dirk von Suchodoletz <dirk@goe.net>, 20-01-2003
#
##############################################################################

# functions
usage() {
cat<<EOM
    mkinittar creates initial tarfile image for booting diskless
linux clients with enviroments which need loading modules before
mounting real root filesystem.
options:
-h This Text.
-dxsrd|-dxsrootdir|-drd
Give the root directory to the DXS installation
-k "kernel" kernel image name (e.g. bzImage)
-d  Copy some binaries for easier debugging into the
   ramdisk
EOM
exit; }
function error () {
    echo -e "->> Error within $0 <<-\n$1\n"
exit 1; }

# interpreting command line options
```

```
(( NARG=0 ))
TEST=( $@ )
while [ $NARG -le $# ] ; do
case "${TEST[$NARG]}" in
*help*|-h)
usage
exit 0
;;
-dxsrd|-dxsrootdir|-drd)
dxsdir=${TEST[$NARG+1]}
(( NARG=$NARG+1 ))
;;
-k)
kernel=${TEST[$NARG+1]}
kernel=`basename $kernel`
(( NARG=$NARG+1 ))
;;
-d)
debug="yes"
esac
(( NARG=$NARG+1 ))
done
# set some default values (this part needs a little bit brush up :-))
if [ x$kernel!="x" ]; then
if [ x$dxsdir!="x" ] ; then
kernel="$dxsdir/boot/$kernel"
else kernel="/nfsroot/dxs/boot/$kernel"
fi
else
kernel="/nfsroot/dxs/boot/bzImage"
fi
[ -f $kernel ] || error "No kernel file could be found. Please specify \
reasonable value\nvia command line options."
[ -d $dxsdir ] || error "No output directory given. It is needed to \
put the tagged\nboot kernel there."

tmp_mnt=/tmp/mnt$$
lx_rc=$tmp_mnt/sbin/init
root_dir=/
cwd=`pwd`

x1=`mount 2>/dev/null | grep "on $root_dir " | tail -1`
x2=`echo \`echo "$x1" | wc -l\``
x3=`echo "$x1" | cut -f 1 -d " "`

exit_code=0

kk="$kernel"
vv=`/sbin/get_kernel_version $kk`
bb=`which busybox`

mkdir $tmp_mnt
cp -a /nfsroot/dxs/var/ram/* $tmp_mnt
cp -a $root_dir/dev/{tty{1,2},zero,null,fb0,console} $tmp_mnt/dev
[ $bb ] || cp /bin/ash $tmp_mnt/bin/sh 2>/dev/null || error 4 "no bash"
touch $tmp_mnt/etc/fstab
cp -a $root_dir/lib/ld-*so* $tmp_mnt/lib
cp -a $root_dir/lib/libc.so.6 $tmp_mnt/lib
cp -a $root_dir/lib/libz.so.1* $tmp_mnt/lib
cp -a $root_dir/lib/libutil.so.1* $tmp_mnt/lib
if [ $bb ] ; then
cp $bb $tmp_mnt/bin
for i in ash cat echo killall sed sh test grep rm cp ; do
ln -s /bin/busybox $tmp_mnt/bin/$i
done
ln -s /bin/test $tmp_mnt/bin/[
else
cp -a $root_dir/bin/{cat,mount,sed,grep,rm,cp}  $tmp_mnt/bin
cp -a $root_dir/usr/bin/killall $tmp_mnt/bin
fi
cp -a $root_dir/bin/mount $tmp_mnt/bin
```

```
cp -a $root_dir/sbin/{dhclient,portmap,ifconfig,route,insmod,modprobe}\
$tmp_mnt/sbin
# programs/files needed for debugging (add your own below)
[ "$debug" ] && cp -a $root_dir/lib/libdl.so.2* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/lib/libhistory.so.4* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/lib/libncurses.so.5* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/lib/libreadline.so.4* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/lib/librt.so.1* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/lib/libpthread.so.0* $tmp_mnt/lib
[ "$debug" ] && cp -a $root_dir/bin/ls $tmp_mnt/bin

strip $tmp_mnt/bin/* $tmp_mnt/sbin/* $tmp_mnt/lib/*.so.* &>/dev/null
# copy needed kernel modules (network adaptor drivers)
mkdir -p $tmp_mnt/lib/modules/$vv/kernel/drivers/{net,pnp}

# translucency
#mkdir -p $tmp_mnt/lib/modules/$vv/kernel/fs
cp -a $dxsdir/lib/modules/$vv/kernel/drivers/net/* \
$tmp_mnt/lib/modules/$vv/kernel/drivers/net
cp $dxsdir/lib/modules/$vv/modules.dep \
$tmp_mnt/lib/modules/$vv
cp -a $dxsdir/lib/modules/$vv/kernel/drivers/pnp/* \
$tmp_mnt/lib/modules/$vv/kernel/drivers/pnp

# translucency
#cp -a $dxsdir/lib/modules/$vv/kernel/fs/translucency.o \

# $tmp_mnt/lib/modules/$vv/kernel/fs
# speed up the boot process but beware: if the filesystem structure of your
# diskless clients differs from your servers it is not a wise idea :-)
cp /etc/ld.so.cache $tmp_mnt/etc

# pack all together
chmod 755 $lx_rc
cd /$tmp_mnt
tar --exclude=inittar.tar.gz -cpzf inittar.tar.gz *
cd $cwd

mv /$tmp_mnt/inittar.tar.gz ./inittar.dxs

# for pxe boot package
cp ./inittar.dxs $dxsdir/boot/inittar.nosplash

# generate splash image for SuSE distribution
#
# change the location of files or add your own bootsplash picture
#
# checking for lpp, if enabled splash is useless
ls -al $dxsdir/boot/bzImage &>/dev/null | grep lpp || {
if test -f /etc/SuSE-release -a -f /bin/splash ; then
VER=‘cat /etc/SuSE-release | grep VERSION | sed -e "s,.*= ,,g"‘
echo "Version: $VER"
case "$VER" in
8.0)
# bootsplash_picture="/usr/share/splash/bootsplash-1024x768.jpg"
# cfgname="/usr/share/splash/bootsplash-1024x768.cfg"
bootsplash_picture="bootsplash-dxs.jpg"
cfgname="bootsplash-dxs.cfg"
source $cfgname
/bin/splash -s -f $cfgname >> ./inittar.dxs
;;
8.1)
# bootsplash_picture="/usr/share/splash/themes/current/images/bootsplash-1024x768.jpg"
# cfgname="/usr/share/splash/themes/current/config/bootsplash-1024x768.cfg"
bootsplash_picture="bootsplash-dxs.jpg"
cfgname="bootsplash-dxs.cfg"
source $cfgname
                /bin/splash -s -f $cfgname >> ./inittar.dxs
# for pxe boot package
cp ./inittar.dxs $dxsdir/boot/inittar.splash
```

```
;;
*)
echo "No clue what to do ..."
;;
esac
else
echo "Could not add splash picture"
fi
}

echo "Merging boot kernel with the ramdisk file"
# make a backup copy
[ -f $dxsdir/boot/bootimg ] && \
cp $dxsdir/boot/bootimg $dxsdir/boot/bootimg.old
mknbi-linux --output=$dxsdir/boot/bootimg \
$dxsdir/boot/bzImage ./inittar.dxs || echo "mknbi-linux not installed"
exit 0
```

Erzeugen der dhcpd.conf erfolgt mittels **dhcp-generate**. Dieses Perl-Skript wurde ursprünglich dazu ausgelegt, die DHCP-Konfiguration anhand von Daten aus einer MySQL-Datenbank zu erstellen. Mit dem Schalter "-i" kann ein Interface für das eine Datei erstellt werden soll, angegeben werden. Der Schalter "-c" schaltet in den Kommand-Line-Modus. Die statischen Vorgaben, welche im Kopf der *dhcpd.conf* landen, sind zu Beginn dieses Skripts festgelegt.

```
#!/usr/bin/perl
#
# script for generating /etc/dhcpd.conf from database output
# or interactive configuration, Version 0.9.5b
#
#
# Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
################################################################

# subroutine for typ cast
sub btodq {
        my $dq = join ".",unpack("CCCC",$_[0]);
        return $dq;
}
sub dqtob {
        my @dq;
        my $q;
        my $i;
        my $bin;

        foreach $q (split /\./,$_[0]) {
                push @dq,$q;
        }
        for ($i = 0; $i < 4 ; $i++) {
                if (! defined $dq[$i]) {
                        push @dq,0;
                }
        }
        $bin    = pack("CCCC",@dq);      # 4 unsigned chars
        return $bin;
}
# subroutine for input
sub input {
my $text=shift;
my $var=shift;
my $empty=shift;
print "   $text [$var]: ";
$input=<STDIN>;
chomp($input);
if ($input ne '') { return $input; }
elsif ($empty) {
print "   Please enter an apropriate value here.\n";
```

```
return (input($text,$var,$empty)); }
return $var;
}
# sub routine for command line parameters
sub cmdline {
my $text=shift;
my $var=shift;
if (($var) && ($var !~ /^-/)) {
return $var; }
else { print $text;
exit 1; }
}

# get command line parameter
$num=0;
while ( $_=shift ) {
# interface declaration
if (/^-ip$/) { $_=shift;
$ip=cmdline("No ip number given!\n",$_);}
elsif (/^-i$/) { $_=shift;
$if[$num]=cmdline("No device name given!\n",$_); $num++;}
elsif (/^-c$/) {
print "Entering interactive mode!!\n\n";
$interactive="y"; }
elsif (/^-h$/) { $_=shift;
$dbhost=cmdline("No host name for database server given!\n",$_);}
elsif (/^-db$/) { $_=shift;
$database=cmdline("No name for database given!\n",$_);}
elsif (/^-u$/) { $_=shift;
$dbuser=cmdline("No user name for database user given!\n",$_);}
elsif (/^-pw$/) { $_=shift;
$dbpw=cmdline("No password to the database given!\n",$_);}
elsif (/^-bc$/) { $_=shift;
$broadcast=cmdline("No value for broadcast given!\n",$_);}
elsif (/^-dg$/) { $_=shift;
$router=cmdline("No value for default gateway given!\n",$_);}
elsif (/^-o$/) { $_=shift;
$outfile=cmdline("No filename for config file given!\n",$_);}
}
if (!$if[0]) {$if[0]="eth0";}

# connect database if not in interactive mode
if (!$interactive) {
use Mysql;
$DB = Mysql->connect($dbhost, $database, $dbuser, $dbpw); }
else {
print "We will now configure your dhcpd settings!\nWe assume you";
print "are running this script on your target host.\n"; }

# open some files
print "OF: $outfile\n";
if (!$outfile) { $outfile="/etc/dhcpd.conf.new"; }
open (OUT, ">$outfile");
open (ERR, ">/var/log/dhcpdconf.log");

# static header of /etc/dhcpd.conf
# some vendor code specific entries made by q&d hack
print OUT "
# /etc/dhcpd.conf
#
# Configuration file for ISC dhcpd
#
# Automatically generated by dhcp-generate.pl\n#\n#";
($sec,$min,$hour,$mday,$mon,$year,
  $wday,$yday,$isday) = localtime (time);
print OUT " --> ",$year+1900,"/",$mon+1,"/",$mday+1;
print OUT " $hour:$min";
print OUT "
#
# (c) Dirk von Suchodoletz <dirk\@goe.net>, 2002
#
```

```
# -- user defined vendor options --

option o128 code 128              = string;
option o129 code 129              = string;
option menudflts code 160         = string;
option motdline1 code 184         = string;
option menuline1 code 192         = string;
option menuline2 code 193         = string;
option menuline3 code 194         = string;
option bootlocal-script code 221  = string;
option language code 222          = string;
option start-x code 223           = string;
option start-snmp code 224        = string;
option start-sshd code 225        = string;
option start-xdmcp code 226       = string;
option start-cron code 227        = string;
option crontab-entries code 228   = string;
option start-rwhod code 229       = string;
option start-printdaemon code 230 = string;
option tex-enable code 232        = string;
option netbios-workgroup code 233 = string;
option vmware code 234            = string;
option hw-mouse code 252          = string;
option hw-graphic code 253        = string;
option hw-monitor code 254        = string;

# -- global options --

option o128                       E4:45:74:68:00:00;
deny                              unknown-clients;
default-lease-time                160000;
max-lease-time                    200000;
use-host-decl-names               on;
option dhcp-max-message-size      1024;
ddns-update-style                 none;

# description for several options:
# A. x-server-defs \"de PS/2 psaux 96 100 1280x1024 nv 16\"
#  - Sets several values for XFree86 V 4.0.X
#
#  1. keyboard  de/us
#  2. mouse typ PS/2/MouseMan/MicroSoft/...
#  3. mouse device psaux/ttyS0X
#  4. horiz. freq. monitor 40 - 96 (depending on monitor)
#  5. vert. freq. monitor 56 -120  (depending on monitor)
#  6. max. resolution
#  6. graphic server module  mga/i810/nv/ati/...
#        (depends on hardware)
#  7. display color depth 8/16/24

# -- vendor identifier dependend settings --
class \"Etherboot\" {
    match if substring (option vendor-class-identifier, 0, 9) = \"Etherboot\";
        option motdline1  = \"Welcome to Guru Labs classroom\";
        option vendor-encapsulated-options 3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
class \"EB-3c905c\" {
    match if substring (option vendor-class-identifier, 0, 9) = \"EB-3c905c\";
#       option menudflts  = \"timeout=30:default=193\";
        option motdline1  = \"Welcome to the Linux PC-Pool\";
        option menuline1  = \"Boot from Hard Drive:::/dev/hda:::\";
        option menuline2  = \"Boot from Network:::/nfsroot/dxs/boot/wsbootimg:::\";
#       option menuline3  = \"Boot from Floppy:::/dev/fd0:::\";
        option vendor-encapsulated-options 3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
class \"SUB-XOpac\" {
    match if substring (option vendor-class-identifier, 0, 9) = \"SUB-XOpac\";
option motdline1  = \"Welcome to SUB-OPAC\";
        option vendor-encapsulated-options 3c:09:53:55:42:ff;
}
class \"PXEClient:\" {
```

```
    match if substring (option vendor-class-identifier, 0, 10) = \"PXEClient:\";
        filename \"/nfsroot/dxs/boot/3c905c-tpo.pxe\";
}

# -- client specific --\n\n";

# dhcp server ip of all installed ethernet interfaces and set corresponding
# network data
$num=0;
while ($if[$num]) {
$iface=$if[$num];
$ipcfg = `/sbin/ifconfig $iface` ;
@ipcfg = split  /^/, $ipcfg ;

foreach $i (@ipcfg) {
$_ = $i ;
if ( /Bcast/ ) {
  @i = split ;
  @j = split ( /\:/, $i[1] ) ;
   if (!$ip) { $ip = $j[1]; }
  @j = split ( /\:/, $i[2] ) ;
   $broadc = $j[1] ;
  @j = split ( /\:/, $i[3] ) ;
   $netm = $j[1] ; }
}

$net = btodq ( dqtob( $ip ) & dqtob( $netm ) );

# write subnet information and server identifier to configuration file
print OUT "subnet $net netmask $netm {\n    server-identifier $ip;\n}\n";

$old_grid=$new_grid=0;

# group hosts and set basic configuration values
if (!$interactive) {
$sth01=$DB->query("select Rechner.GroupingID,
  HostName, IPAddress, RechnerID from Rechner, Grouping_Properties
  where Rechner.GroupingID=Grouping_Properties.GroupingID
  AND PropertyNameID=34 AND Value='$ip' order by
  Grouping_Properties.GroupingID, HostName;");
  $bla="$sth01->fetchrow"; }
else { $data[0]=1; }

while ( ($interactive)||(@data = $sth01->fetchrow) )
{
if ($interactive eq 'n') { last; }
    $new_grid=$data[0];
print "\nRunning host group loop\n\n";
$hostname=$data[1];
    if ( $old_grid != $new_grid ) {
  if ( $old_grid != 0 ) { print OUT "}\n"; }
# get the group data of the computer
if (!$interactive) {
  $sth02=$DB->query("SELECT FirstDNS, SecondDNS, DomainName, Broadcast,
  GateWay, NetMask, Name, RechnerArtID FROM Grouping WHERE
  GroupingID=$new_grid");
  @data02 = $sth02->fetchrow; }
else {
  print "Just press 'Enter' if you like to take the default value.\n";
  print "Empty hostgroup finishes the dhcpd configuration process.\n";
  $data02[6]=input("Please enter name of hostgroup","");
  if ( $data02[6] ne ""  ) {
$domain_name_servers=input("Enter comma separated list of name server ip's",$domain_name_servers);
$broadcast=$broadc;
$broadcast=input("Enter broadcast address",$broadcast);
if (!$router) { $router='0.0.0.0'; }
$router=input("Enter default router",$router);
$domain_name=input("Enter domain name",$domain_name); }
# finish loop if no hostgroup defined
  else { last; } }
  if ( $data02[0] ne '' ) { $domain_name_servers=$data02[0]; }
```

```
  if ( $data02[1] ne '' ) { $domain_name_servers.=', '.$data02[1]; }
  if ( $data02[2] ne '' ) { $domain_name=$data02[2]; }
  if ( $data02[3] ne '' ) { $broadcast=$data02[3]; }
  if ( $data02[4] ne '' ) { $router=$data02[4]; }
  if ( $data02[7] ne '1' ) { $filename="/nfsroot/dxt/boot/bootimg";
            $rootpath="/nfsroot/dxt"; }
  else { $filename="/nfsroot/dxs/boot/wsbootimg";
        $rootpath="/nfsroot/dxs"; }

# fetching further group data
if (!$interactive) {
  $sth03=$DB->query("SELECT Grouping_Properties.Value, PropertyNamen.Name
  FROM Grouping_Properties, PropertyNamen WHERE
  Grouping_Properties.GroupingID='$new_grid' AND
  Grouping_Properties.PropertyNameID=PropertyNamen.PropertyNameID");
  while ( @add_data = $sth03->fetchrow )
  {
SWITCH: {
( $add_data[1] eq 'XDM' ) && do { $start_xdmcp=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'RWHO' ) && do { $start_rwho=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'CRON' ) && do { $start_cron=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'X11' ) && do { $start_x11=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'SNMP' ) && do { $start_snmp=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'DNETC' ) && do { $start_dnetc=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'NIS-Domain' ) && do { $nis_domain=$add_data[0];
   last SWITCH; };
( $add_data[1] eq 'X-Login-Server' ) && do {
if ( $x_display_manager eq '' )
{ $x_display_manager=$add_data[0]; }
else
{ $x_display_manager.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'NTP-Server' ) && do {
if ( $ntp_servers eq '' )
{ $ntp_servers=$add_data[0]; }
else
{ $ntp_servers.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'Log-Server' ) && do {
if ( $log_servers eq '' )
{ $log_servers=$add_data[0]; }
else
{ $log_servers.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'LPR-Server' ) && do {
if ( $lpr_servers eq '' )
{ $lpr_servers=$add_data[0]; }
else
{ $lpr_servers.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'Font-Server' ) && do {
if ( $font_servers eq '' )
{ $font_servers=$add_data[0]; }
else
{ $font_servers.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'DomainSearch' ) && do {
if ( $domain_name eq '' )
{ $domain_name=$add_data[0]; }
else
{ $domain_name.=' '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'NIS-Server' ) && do {
if ( $nis_servers eq '' )
{ $nis_servers=$add_data[0]; }
```

```
else
{ $nis_servers.=', '.$add_data[0]; }
last SWITCH; };
}
  }
} else {
print "Please enter additional data for this hostgroup!\n";
print "Lists (valid only for some variables) should be\n";
print "entered comma separated!\n";
if (!$start_xdmcp) { $start_xdmcp='yes'; }
$start_xdmcp=input("Start XDM (yes/no)           ",$start_xdmcp);
if (!$start_rwho) { $start_rwho='yes'; }
$start_rwho=input("Start RWHO daemon            ",$start_rwho);
if (!$start_cron) { $start_cron='yes'; }
$start_cron=input("Start CRON  daemon           ",$start_cron);
if (!$start_x11) { $start_x11='yes'; }
$start_x11=input("Start X11 graphical display  ",$start_x11);
if (!$start_snmp) { $start_snmp='yes';}
$start_snmp=input("Start SNMP daemon            ",$start_snmp);
if (!$start_dnetcp) { $start_dnetc='no'; }
$start_dnetc=input("Start DNETC service          ",$start_dnetc);
$nis_domain=input("Enter NIS-Domain             ",$nis_domain);
if ( $nis_domain ne '' ) {
$nis_servers=input("Enter list of NIS-Servers    ",$nis_servers); }
$x_display_manager=input("Enter list of X-Login-Servers",$x_display_manager);
}
print OUT "#\n# Rechner -> $data02[6]\n#\n";
print OUT "group {\n";
print OUT "    filename \"$filename\";\n";
print OUT "    option root-path \"$rootpath\";\n";
if ( $broadcast ne '' ) {
print OUT "    option broadcast-address $broadcast;\n"; }
else {
   print ERR "Warn: Gruppe $data02[6] kennt keine Broadcast-Adresse\n"; }
if ( $router ne '' ) {
print OUT "    option routers $router;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine Gateway-Adresse\n"; }
if ( $domain_name_servers ne '' ) {
print OUT "    option domain-name-servers $domain_name_servers;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine DNS-Adresse(n)\n"; }
if ( $domain_name ne '' ) {
print OUT "    option domain-name \"$domain_name\";\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine Domain(s)\n"; }
if ( $nis_domain ne '' ) {
print OUT "    option nis-domain \"$nis_domain\";\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine NIS-Domain\n"; }
if ( $nis_servers ne '' ) {
print OUT "    option nis-servers $nis_servers;\n"; }
else {
   if ( $nis_domain ne '' ) {
  print ERR "Crit: Gruppe $data02[6] kennt keine NIS-Server\n";
      print ERR "  hat aber die NIS-Domain $nis_domain definiert\n"; }
   else {
  print ERR "Info: Gruppe $data02[6] kennt keine NIS-Server\n"; } }
if ( $log_servers ne '' ) {
print OUT "    option log-servers $log_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine Log-Server\n"; }
if ( $ntp_servers ne '' ) {
                print OUT "    option ntp-servers $ntp_servers;\n"; }
        else {
            print ERR "Info: Gruppe $data02[6] kennt keine NTP-Server\n"; }
if ( $lpr_servers ne '' ) {
print OUT "    option lpr-servers $lpr_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine LPR-Server\n"; }
if ( $font_servers ne '' ) {
```

```
print OUT "    option font-servers $font_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine Font-Server\n"; }
if ( $x_display_manager ne '' ) {
print OUT "    option x-display-manager $x_display_manager;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt XDMCP-Server\n"; }
if ( $start_x11 ne '' ) {
print OUT "    option start-x \"$start_x11\";\n"; }
else {
    print ERR "Warn: In Gruppe $data02[6] fehlt der X-Zugriffsmodus\n"; }
if ( $start_xdmcp ne '' ) {
print OUT "    option start-xdmcp \"$start_xdmcp\";\n"; }
else {
    print ERR "Info: In Gruppe $data02[6] wird auf xdmcp verzichtet\n"; }
print OUT "    option start-rwhod \"$start_rwho\";\n";
print OUT "    option start-cron \"$start_cron\";\n";
print OUT "    option start-snmp \"$start_snmp\";\n";
print OUT "    option start-dnetc \"$start_dnetc\";\n    #\n";
# reset of variables
if (!$interactive) {
$nis_domain=$nis_servers=$font_servers=$log_servers='';
$x_display_manager=$boot_local=$mac=$ntp_servers='';
$start_dnetc=$start_snmp=$start_cron=$start_rwho=$start_xdmcp=''; }
# end of newgid != oldgid
  }
# get keyboard language
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
   HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
   HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
       HardWareNamen.HardWareArtID=3 AND
   HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
   AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=53");
if ( @add_data = $sth2->fetchrow ) { $key_lang = $add_data[0]; } }
else {
# in interactive mode ask for hostname and ip before asking hw questions
print "\n--- Now in host entry loop (group: ".$data02[6].")";
print " ---\nYou have to provide some host specific data!\n";
$data[1]=input("Enter Hostname","","s");
$data[2]=input("Enter IP number of this host",$data[2],"ip");
print "We need now hardware specific information starting with\n";
if (!$key_lang) { $key_lang='de'; }
$key_lang=input("keyboard language",$key_lang); }
if  ( $key_lang eq '' ) {
    print ERR "Warn: Setze fuer $data[1] Tastatur-Default-";
print ERR "Sprache auf Deutsch\n";
$key_lang = "de"; }

# get the mouse protocol
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
   HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
   HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
       HardWareNamen.HardWareArtID=2 AND
   HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
   AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=56");
   if ( @add_data = $sth2->fetchrow ) { $mp = $add_data[0]; } }
else {
if (!$mp) { $mp='imps/2'; }
$mp=input("Enter mouse protocol (ps/2, imps/2, mouseman ...)",$mp); }
if  ($mp eq '') {
print ERR "Crit: Setze fuer $data[1] Maus-Protokoll auf";
print ERR "PS/2 da nicht in Datenbank\n";
$mp = "PS/2"; }

# get the mouse port
if (!$interactive) {
$sth2=$DB->query("SELECT PropertyNamen.Name FROM
   HardWareNamen_Properties, HardWareNamen, HardWare, PropertyNamen
   WHERE PropertyNamen.PropertyArtID=3 AND
```

```
   PropertyNamen.PropertyNameID=
   HardWareNamen_Properties.HardWareNamen_PropertyNameID AND
   HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
   AND HardWareNamen.HardWareArtID=2 and HardWare.RechnerID=$data[3]
   AND HardWare.HardWareNameID=HardWareNamen.HardWareNameID");
if ( @add_data = $sth2->fetchrow ) { $_ = $add_data[0];
  if (/Serial/) { $md = "ttyS0"; } else { $md = "psaux"; } } }
else {
if (($mp eq 'imps/2')||($mp eq 'ps/2')) { $md='psaux'; }
else { $md='ttyS0'; }
$md=input("Enter mouse device (tty0, psaux, ...)",$md); }
if  ($md eq '') {
print ERR "Crit: Setze fuer $data[1] Maus-Anschluss auf";
print ERR "psaux da nicht in Datenbank\n";
$md = "psaux"; }

# get monitor data
# 1) max. screen resolution
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
   HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
   HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
       HardWareNamen.HardWareArtID=1 AND
   HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
   AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=2");
if ( @add_data = $sth2->fetchrow ) { $max_res = $add_data[0]; } }
else {
if (!$max_res) { $max_res='1024x768'; }
$max_res=input("Enter maximum resolution for the connected monitor\n\
\t1024x768, 1280x1024, ...)",$max_res); }
if ($max_res eq '') {
    print ERR "Warn: Setze fuer $data[1] Default-Monitor-";
print ERR "Aufloesung auf 1024x768 da nicht in Datenbank\n";
$max_res = "1024x768"; }

# 2) max. horizontal freq.
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
     HardWareNamen.HardWareArtID=1 AND
HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=14");
if ( @add_data = $sth2->fetchrow ) { $hfreq = $add_data[0]; } }
else {
$hfreq=input("Enter max. horizontal sync (kHz).",$hfreq,"n"); }
if ($hfreq eq '') {
print ERR "Crit: setting for $data[1] monitor vertical refresh";
print ERR "to 65kHz \n";
$hfreq = "65"; }

# 3) max. vertical freq.
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
     HardWareNamen.HardWareArtID=1 AND
HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=13");
if ( @add_data = $sth2->fetchrow ) { $vfreq = $add_data[0]; } }
else {
$vfreq=input("Enter max. vertical refresh (Hz)",$vfreq); }
if ($vfreq eq '') {
    print ERR "Crit: setting for $data[1] monitors vertical refresh";
print ERR "to 90Hz ...\n";
$vfreq = "90"; }

# 4) color depth (should be computed from resolution and amount of graphic memory)
$color_d="16";

# 5) XFree86 driver (Version 4.X.Y)
```

```
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
(HardWareNamen.HardWareArtID=5 OR
HardWareNamen.HardWareArtID= 23) AND
HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=10");
if ( @add_data = $sth2->fetchrow ) { $driver = $add_data[0]; }
else {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
   HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
   HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
   (HardWareNamen.HardWareArtID=5 OR
   HardWareNamen.HardWareArtID= 23) AND
   HardWareNamen_Properties.HardWareNameID=HardWareNamen.HardWareNameID
   AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=11");
if ( @add_data = $sth2->fetchrow ) { $driver = $add_data[0]; } } }
else {
$driver=input("Enter the XFree module name for the graphics board",$driver); }
if ($driver eq '') {
print ERR "Warn: Setze fuer $data[1] XFree86-Modul auf";
print ERR "'vesa' (generic)\n";
    $driver = 'vesa'; }
# get the MAC address
if (!$interactive) {
$sth2= $DB->query("SELECT Value FROM HardWare, HardWare_Properties,
HardWareNamen where HardWare.RechnerID=$data[3] AND
HardWare.HardWareID=HardWare_Properties.HardWareID AND
HardWare.HardWareNameID=HardWareNamen.HardWareNameID AND
HardWare_Properties.PropertyNameID=44");
if ( @add_data = $sth2->fetchrow ) { $mac = $add_data[0]; } }
else {
$mac="01:02:03:04:05:06";
$mac=input("Enter the MAC address of the network\n\tadaptor",$mac); }
if ($mac eq '') {
    print ERR "Crit: Keine Mac-Adresse fuer Netzwerkkarte in";
print ERR "$data[1] ermittelbar\n";
$mac = "00:00:00:00:00:00"; }

# get needed kernel modules from database
if (!$interactive) {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen_Properties
  WHERE HardWare.RechnerID='$data[3]' AND
  HardWareNamen_Properties.HardWareNameID=HardWare.HardWareNameID
  AND HardWareNamen_Properties.HardWareNamen_PropertyNameID='15'");

while ( @add_data = $sth2->fetchrow )
    {
if ( $boot_local eq '' ) {
    $boot_local='modprobe -qa '.$add_data[0]; }
else {
    $boot_local.=' '.$add_data[0]; }
} }
else {
$boot_local=input("list of kernel modules to load manually",$boot_local); }

# fill up the host part of the dhcpd.conf
print OUT "    host $data[1] {\n";
print OUT "\thardware ethernet $mac;\n";
print OUT "\toption x-server-defs \"$key_lang $mp $md $hfreq ";
print OUT "$vfreq $max_res $driver $color_d\";\n";
print OUT "\toption bootlocal-script \"$boot_local\";\n";
# switch on framebuffer & lpp for distinct type of graphic boards
if ( $driver eq 'ati' || $driver eq 'nv' || $driver eq 'XF86_S3' ) {
    print OUT "\toption o129 \"vga=0x0301 console=/dev/tty2 ";
print OUT "CONSOLE=/dev/tty2\";\n"; }
print OUT "\tfixed-address $data[2];\n    }\n";

# reset of variables
$key_lang=$mac=$mp=$md=$vfreq=$hfreq=$max_res=$driver=$color_d="";
```

```
$boot_local="";

# check for new group
if ($interactive) {
print "\n\nNew group? ";
print "('enter' keeps the current one: \"";
print $data02[6]."\")\nThe entry of 'end' stops the input loop: ";
$input=<STDIN>; chomp $input;
if ( $input eq 'end' ) { last; }
elsif ( $input ne "" ) { $new_grid+=1; $data02[6]=''; } }

# group order
    $old_grid=$new_grid;
}
# closing
if (($old_grid) || ($input eq 'end')) { print OUT "}\n"; }
# closing of the input loop
$num++;
}

# close files
close (OUT);
close (ERR);
```

B.3 Laufzeitskripten

Start aus der Ramdisk: init Das erste Programm, welches der Linuxkernel ausführt, ist **init**. Dieses ist im INITTAR ein Shell-Skript für die **ash** (bzw. **busybox** mit einem Ash-Link). Nacheinander wird das Netzwerkkartenmodul geladen und das Netzwerkinterface mittels **dhclient** und **dhclient-script** konfiguriert. Nachdem **dhclient-script** sich beendet hat, führt **init** das **pivot_root** durch und beendet sich durch den Aufruf des "richtigen" **init**. Um die konfigurierten Debug-Meldungen anzuzeigen, muss in der Kernel-Commandline ein Token, wie "debug=N" stehen, wobei *0* für keinen Output, *1* für mäßigen und *2* für maximalen Output definiert wurden. Sonst kennt das **init**-Sktipt keine weiteren Optionen.

```
#!/bin/sh
#
# Setup script for linux diskless clients inside an inittar environment
# 1) check for network adapter (modprobing known ethernet adaptors)
# 2) get and set configuration parameters via dhclient
# 3) switch to "normal" init after root filesystem is installed
#
# Copyright (c) 2003, Version: 0.9b
#
# Author: Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
# /sbin/init (-> from dxs/var/ram/sbin/init via mkdxsinittar)

export PATH=/bin:/usr/bin/:/sbin:/mnt/bin:/mnt/sbin:/mnt/usr/bin

# functions
error () {
echo -e "->> Error <<-\n$1\n"
if [ -f /bin/busybox ] || [ -f /RAM/bin/busybox ] ; then
echo -e "\tIf you need some more commands for debugging link them"
echo -e "\tagainst busybox. Most of standard file commands are"
echo -e "\tavailable that way."
else
echo -e "\tIf you need some more commands for debugging run the 'mkinittar'"
echo -e "\tscript with the '-debug' option. It will add some useful"
echo -e "\tcommands but increases your INITTAR file size."
fi
echo
echo -e "\tStarting interactive subshell for problem investigation now..."
echo -e "\t->> If you dont like to have this shell started for security"
echo -e "\treasons, comment this feature in the error function out."
export PS1="debug # "
```

```
exec /bin/sh
exit 1
}

# proc fs and devfs are needed for some tasks
mount -n -t proc proc /proc

# check if debug flag is set via kernel command line
grep -i debug /proc/cmdline >/dev/null 2>&1 && \
DEBUG=‘cat /proc/cmdline|sed -e "s,.*debug,," -e "s,=,," -e "s, .*,,"‘
[ $DEBUG ] || DEBUG=0
msg="Running INITTAR environment ..."
if [ -f /proc/progress ] ; then
    echo "10 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
echo " " >/dev/null 2>&1 || error "The device filesystem is not accessable! \
Check if it is mouonted correctly\nvia kernel options. Or rewrite the init \
script for operation without 'devfs'!"

# work in quiet mode (less debugging output)
[ $DEBUG -lt 2 ] && echo "0 0 0 0" >/proc/sys/kernel/printk

# configure loopback device for portmap and mounting
msg="Setting up basic networking ..."
if [ -f /proc/progress ] ; then
echo "12 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
ifconfig lo 127.0.0.1 netmask 255.0.0.0 up || \
error "'ifconfig' command not found or some similar error occured."
route add -net 127.0.0.0 netmask 255.0.0.0 dev lo || \
error "'route' command failed or command not fount."

# start portmapper, needed for nfs mounting
msg="Starting portmapper ..."
if [ -f /proc/progress ] ; then
echo "14 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
portmap || error "'portmap' failed. It is strongly recommended if you \
try to mount\nnfs shares."

# try to load a module and stop the process if one acceptable was found
# start with the most probable ones in your pool of linux diskless clients
msg="Checking for ethernet interface ..."
if [ -f /proc/progress ] ; then
echo "16 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
for module in 3c59x \
8139too tulip natsemi sis900 \
via-rhine ne2k-pci dmfe epic100 \
winbond-840 eepro100 e2100 cs89x0 \
bcm4400 tg3 e1000 \
lance pcnet32 smc9194 3c509 \
wd smc-ultra eexpress ni65 \
"ne io=0x300" "ne io=0x320" "ne io=0x340" \
"ne io=0x360" "ne io=0x200" "ne io=0x220" \
"ne io=0x240" "ne io=0x250" "ne io=0x260" \
"ne io=0x280" \
depca eepro ni5010 ni52 \
3c515 3c507; do
# to try for more adaptores comment the 'break' command out
[ $DEBUG -gt 0 ] && echo "Trying module: $module"
modprobe -q $module >/dev/null 2>&1 && break
done
```

```
# power up the ethernet interface
msg="Configuring ethernet interface ..."
if [ -f /proc/progress ] ; then
echo "18 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
ifconfig eth0 up || \
error "No network interface card could be detected! Check the kernel \
modules\ninstalled to INITTAR and isa nic adapter list in init script. \
The booted\nkernel MUST match the modules within INITTAR and the modules.\
conf should\nbe up to date."
# get configuration information via dhcp
msg="Getting Configuration Info via DHCP ..."
if [ -f /proc/progress ] ; then
echo "20 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
killall dhclient >/dev/null 2>&1 && \
error "Dhclient process is looping - stopping now."
dhclient -1 -lf /run/dhclient.leases -pf /run/dhclient.pid eth0 \
>/dev/null 2>&1 || error "Problems running dhclient and getting a lease \
(useable and complete ip configuration)! Check if all requested variables \
(see /etc/dhclient.conf) are presented by the server. Check the \
(/var)/log/boot.log file too."

# check the mounted filesystems
msg="Got configuration - checking it now ..."
if [ -f /proc/progress ] ; then
echo "22 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
mount | grep nfs >/dev/null 2>&1 || \
error "No nfs filesystem seems to be mounted at this stage. See if your \
network\ninterface was configured correctly. Simply type 'ifconfig' at \
the\nprompt for checking. Maybe more variables were declared as 'required'\
\nthan your server is offering. Check dhclient.conf with 'cat \
/etc/dhclient.conf'."

# stop dhclient and portmap (portmap prevents /proc from unmounting)
killall dhclient portmap >/dev/null 2>&1

# reorganize the root file system
msg="Reorganizing root filesystem ..."
if [ -f /proc/progress ] ; then
echo "24 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
cd /mnt
umount /proc
pivot_root . RAM || \
error "Could not execute 'pivot_root' due to missing command or wrong \
parameters\ngiven."
cd /

# read mount information from our new root fs and check for the existance
# of important fs (procfs and devfs)
mount -an >/RAM/dev/null 2>&1
mount | grep devfs >/dev/null 2>&1 || \
error "Device filesystem (devfs) should be mounted now or system will \
not boot\nup correctly. Check  (var/ram/ or RAM)/etc/fstab file too."
mount | grep proc >/dev/null 2>&1 || \
error "Process filesystem (proc) should be mounted now or system will \
not boot\nup correctly."

# clean up and free most of the tmpfs memory (binaries avail. via nfs now)
# these files and directories not needed any further
rm -rf /RAM/bin /RAM/sbin /RAM/lib /RAM/proc >/dev/null 2>&1
```

```
rm -rf /RAM/dev /RAM/etc/*.default /RAM/etc/samba/*.default \
/RAM/mnt >/dev/null 2>&1

# reenable the kernel messaging system
echo "1 4 1 7" >/proc/sys/kernel/printk

# fire up our client from nfs root fs (and exit)
msg="Running real init ..."
if [ -f /proc/progress ] ; then
echo "26 "$msg >/proc/progress
else
    [ $DEBUG -gt 0 ] && echo $msg
fi
exec /sbin/init || \
error "Could not invoke '/sbin/init' correctly. Please check if root \
filesystem\nwas mounted correctly and boot parameters are OK."
```

Die Konfiguration mittels dhclient-script Alle wesentlichen Variablenfestlegungen, die keine Hardwareerkennung erfordern, erfolgen durch **dhclient-script**. Dieses erhält seine Informationen per Kommandozeile von **dhclient**, welches wiederum in seinem Verhalten durch *(/var/ram)/etc/dhclient.conf* bestimmt wird. **dhclient-script** ist ein **ash**-Skript.

```
#!/bin/sh
#
# /sbin/dhclient-script -> for linux diskless clients
#
# Version: 0.4.5a
#
# Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
##############################################################################

: ${failsafe="echo 'error occured with dhclient-script; exit 0"}
readonly failsafe
trap "exec $failsafe" EXIT SIGHUP SIGINT SIGPIPE SIGTERM SIGIO

# all major configuration and setup tasks are done within this script
# a variety of services is configured within here - look for config.default
# files in -nfsroot- /var/ram/etc* directory
#
# 1. setup network interface(s)

# 2. mount filesystem structure via nfs
# a) root filesystem consisting of the basic directory structure
# b) add at least /opt and /usr directories for applications
# c) eventually add /tmp/scratch (or similar) to save space in tmpfs

# 3. set up the other configuration files

# configure dxs debug level and set logfile name
grep -i debug /proc/cmdline >/dev/null 2>&1 && \
DXSDEBUG=`cat /proc/cmdline|sed -e "s,.*debug,," -e "s,=,," -e "s, .*,,"`
[ $DXSDEBUG ] || DXSDEBUG=1
if [ "$DXSDEBUG" != "0" ]; then
LOGFILE="/log/boot.log"
else
   LOGFILE="/dev/null"
fi

# write info to log file
echo -e "Running $0 ...\n" >>$LOGFILE

# it is no sense to use debug level higher then "1" with lpp enabled
if ([ -w /proc/progress ]  && [ $DXSDEBUG -gt 1 ]) ; then
DXSDEBUG="1"
echo -e "$0: It does not make sense to use debug level higher \
then 1\n if using linux progress patch (lpp). Switching to 1." >>$LOGFILE
echo -e "If the progress patch seems to show weird colors or \
font settings\ncheck your dhcpd.conf for the correct vga=0x301 settings. \
```

```
These settings\nare passed to the kernel via kernel command line." \
>>$LOGFILE
fi

if [ x$new_broadcast_address != x ]; then
  new_broadcast_arg="broadcast $new_broadcast_address"
fi
if [ x$new_subnet_mask != x ]; then
  new_subnet_arg="netmask $new_subnet_mask"
fi

ifconfig $interface inet $new_ip_address $new_subnet_arg   \
$new_broadcast_arg
# get network address name
netaddr=`route -n|grep "eth0"|sed -e "s, .*,,g"`

# add a network route to the computed network address
for router in $new_routers; do
route add default gw $router
done

if [ x$new_ip_address != x$alias_ip_address ] && \
[ x$alias_ip_address != x ]; then
ifconfig $interface:0- inet 0
ifconfig $interface:0 inet $alias_ip_address $alias_subnet_arg
route add -host $alias_ip_address $interface:0
fi

# it is time to mount the nfs root filesystem now
if [ x$new_root_path != x ] && [ x$new_dhcp_server_identifier != x ]; then \
mount -n -t nfs -o ro,nolock,rsize=8192,intr \
$new_dhcp_server_identifier:$new_root_path /mnt || \
echo -e "$0: Error mounting root filesystem via nfs. Check correctness \
of the \$new_dhcp_server_identifier (value: $new_dhcp_server_identifier) and \
\$new_root_path (value: $new_root_path) variables (dhcpd.conf)!" >>$LOGFILE
mount -n -t nfs -o rw,nolock,rsize=8192,wsize=8192,intr \
$new_dhcp_server_identifier:/tmp/dxs /tmp/scratch || \
echo -e "$0: Error mounting /tmp/dxs filesystem via nfs." >>$LOGFILE
mount -n -t nfs -o ro,nolock,rsize=8192,intr \
$new_dhcp_server_identifier:/opt /mnt/opt || \
echo -e "$0: Error mounting /opt filesystem via nfs." >>$LOGFILE
mount -n -t nfs -o ro,nolock,rsize=8192,intr \
$new_dhcp_server_identifier:/usr /mnt/usr || \
echo -e "$0: Error mounting /usr filesystem via nfs." >>$LOGFILE
else
echo -e "$0: Check that the variables new_dhcp_server_identifier (value: \
$new_dhcp_server_identifier) and new_root_path (value: $new_root_path) are defined \
within the dhcpd.conf. They are needed for mounting!" >>$LOGFILE
fi

# set tex environment if requested
if [ x$new_tex_enable != x ] && [ x$new_tex_enable != xno ] && \
[ x$new_dhcp_server_identifier != x ]; then \
mount -n -t nfs -o ro,nolock,rsize=8192,intr \
$new_dhcp_server_identifier:/var/lib/texmf /mnt/var/lib/texmf || \
echo -e "$0: Error mounting tex /var/lib/texmf via nfs." >>$LOGFILE
mount -n -t nfs -o rw,nolock,rsize=8192,wsize=8192,intr \
$new_dhcp_server_identifier:/var/cache/fonts /mnt/var/cache/fonts || \
echo -e "$0: Error mounting tex fontcache via nfs." >>$LOGFILE
fi

# set new hostname
test -n "$new_host_name" && \
{ echo $new_host_name >/proc/sys/kernel/hostname ;
echo >/etc/hosts
sed -e "s,#1.2.3.4*,$new_ip_address,g" \
    -e "s,default,$new_host_name,g" /etc/hosts.default \
    >>/etc/hosts; }

# write font server ip to file
test -n "$new_font_servers" && { echo > /etc/FONTSERVER; \
```

```
for fontserver in $new_font_servers; do
echo $new_font_servers >>/etc/FONTSERVER;
    done; }

# configure print server (lpd or cups)
if [ "x$new_start_printdaemon" != "x" ] && \
[ "x$new_start_printdaemon" != "xno" ]; then
case $new_start_printdaemon in
    yes|cups*|CUPS*)
start_lpd="no"
start_cups="yes"
    ;;
    lp*|LP*|PLP*)
start_lpd="no"
start_cups="yes"
    ;;
    *)
echo -e "$0: Do not know print-server \
 $new_start_printdaemon; specify 'cups' or 'lpd'." >>$LOGFILE
    ;;
esac
fi

# configure syslog daemon
test -n "$new_log_servers" || new_log_servers="127.0.0.1"
echo >/etc/syslog.conf;  \
for logserver in $new_log_servers; do
sed -e "s,#\*.\*;*\*.*,*.*;*kern.!*    @$logserver,g"  \
/etc/syslog.conf.default >>/etc/syslog.conf
done


# set up nis if defined and add nis '+' to the passwd file. If you like to
# use nis for other configuration files too, add them to the ramdisk. Add
# the appropriate lines if needed.
start_nis=no;
if [ "x$new_nis_domain"!="x" ] && [ "x$new_nis_servers"!="x" ]
then  echo $new_nis_domain >/proc/sys/kernel/domainname
  echo ypserver $new_nis_servers >/etc/yp.conf
  echo "+::::::" >>/etc/passwd
  start_nis=yes;
fi

# write available X display manager to var/lib/xdm/Xaccess
test -n "$new_x_display_manager" && \
echo -e "*\n%hostlist\t$new_x_display_manager" >/etc/X11/Xaccess
echo -e "*\t\tCHOOSER %hostlist" >>/etc/X11/Xaccess

# set up domainname and resolving
test -n "$new_domain_name" && \
echo search $new_domain_name >/etc/resolv.conf
test -n "$new_domain_name_servers" && {
for nameserver in $new_domain_name_servers; do
echo nameserver $nameserver >>/etc/resolv.conf;
done; }

# set up samba configuration with NetBIOS name servers etc.
if [ "x$new_netbios_name_servers" != "x" ]; then
wins_server="\\twins server = $new_netbios_name_servers"
else
wins_server=";\\twins server = 1.2.3.4"
fi
if [ "x$new_netbios_workgroup" != "x" ]; then
workgroup="workgroup = $new_netbios_workgroup"
else
workgroup="workgroup = DXS-DEFAULT"
fi

test -f /etc/samba/smb.conf.default && \
sed -e "s,\s*netbios name.*,netbios name = $new_host_name,g" \
-e "s,\s*interfaces.*,interfaces = \
```

```
$new_ip_address/$new_broadcast_address,g" \
-e "s,;.*wins server.*,$wins_server,g"   \
-e "s,\s*workgroup =.*,$workgroup,g" \
    /etc/samba/smb.conf.default | grep -v "#" >/etc/samba/smb.conf;
test -f /etc/lisarc.default && \
sed -e "s,AllowedA.*,AllowedAddresses=\
$new_ip_address/$new_subnet_mask;,g" \
-e "s,BroadcastN.*,BroadcastNetwork=\
$new_ip_address/$new_subnet_mask;,g" \
-e "s,PingAdd.*,PingAddresses=\
$new_ip_address/$new_subnet_mask;,g" \
/etc/lisarc.default >/etc/lisarc;

# set up net time service configuration
test -n "$new_ntp_servers" && { echo >/etc/ntp.conf;
for ntpserver in $new_ntp_servers; do
initntp="$initntp $ntpserver"
echo server $ntpserver >>/etc/ntp.conf;
done;
start_xntp=yes; }
echo "marke 2g"
# set up xinetd service configuration
test -f /etc/xinetd.conf.default && \
sed -e "s,\s*only_from.*,only_from       = \
$netaddr/$new_subnet_mask,g" \
/etc/xinetd.conf.default >/etc/xinetd.conf

# set up snmpd configuration
test -n "$new_start_snmp" && \
sed -e "s,NETADDR/MASK,$netaddr/$new_subnet_mask,g" \
/etc/ucdsnmpd.conf.default >/etc/ucdsnmpd.conf

#set up printing environment
test -n "$new_lpr_servers" && \
    echo -e "ServerName $new_lpr_servers" >/etc/client.conf

#    lpstat -a -h $new_lpr_servers | \
#    awk '{ print $1":"}' >/RAM/etc/printcap & )

# create inittab (decide according to x directives)
defaultwm=kde
if [ -f /proc/progress ]; then
vt="vt1";
else vt="vt5";
fi
init="#7:5:respawn:/usr/X11R6/bin/X $vt"
case $new_start_x in
no|none|NO)
init="#$init -query localhost"
;;
broadcast|BROADCAST)
init="$init -broadcast"
;;
indirect|INDIRECT)
[ $new_host_name ] || new_host_name="localhost"
init="$init -indirect $new_host_name"
echo -e "$0: Don't forget to enable xdmcp within your \
displaymanagers configuration\nfile: xdm-config for xdm, kdmrc for kdm, \
gdmrc for gdm ..." >>$LOGFILE;
;;
query|QUERY)
init="$init -query localhost"
;;
# vmware|VMware|VMWARE)
# [ x$new_start_vmkernel != "xyes" ] && new_start_vmkernel="yes"
# [ x$new_start_xdmcp != "x" ] || new_start_xdmcp="no"
# init="#7:5:respawn:su -c 'xinit /sbin/startgui "
# init=$init"vmware \&>/dev/null' -l nobody"
# ;;
*)
[ x$new_start_xdmcp != "x" ] || new_start_xdmcp="no";
```

```
defaultwm=$new_start_x
init="#7:5:respawn:su -c 'xinit /sbin/startgui "
init=$init"$new_start_x \&>/dev/null'"
init="$init -l nobody"
echo -e "$0: If you dont like to have X without login \
started,\nset variable 'start-x' via dhcp to the appropriate \
value." >>$LOGFILE
;;
esac
echo >/etc/inittab
sed -e "s,#7:5.*,$init \&>/dev/null,g" /etc/inittab.default >>/etc/inittab

# vmware
[ x$new_start_x = x"vmware" ] && start_vmkernel="yes"
[ x$new_vmware != x ] && [ x$new_vmware != xno ] && start_vmkernel="yes"

# add commands to boot.local
test -n "$new_bootlocal_script" && \
echo $new_bootlocal_script >>/run/boot.local

# set up /etc/rc.config (SuSE linux < 8.0)
# all parameters must be present, at least via defaults in
# (nfsroot/dxs)/etc/dhclient.conf
#
# do some checks on the variables at first
[ x$new_start_xdmcp != "xkdm" ] && [ x$new_start_xdmcp != "xgdm" ] && \
[ x$new_start_xdmcp != "xxdm" ] && [ x$new_start_xdmcp != "xwdm" ] && \
[ x$new_start_xdmcp != "xno"  ] && \
{ new_start_xdmcp="kdm"; echo -e "$0: No variable was set for \
x display manager - setting it to 'kdm'." >>$LOGFILE; }

# try to configure gpm for console mouse
start_gpm=no
[ "$new_hw_mouse" ] && for token in $new_new_hw_mouse; do
case $token in
tty*|psaux|usb) MD=$token
;;
*ps/2|Mouse*|Micr*) MP=`echo $token | sed -e "s,/,,g"`
;;
esac
done
[ "$MD" ] && [ "$MP" ] && start_gpm=yes

# try to gather language variable for keyboard and system environment
KEYTABLE=`echo $new_language|sed -e 's, .*,,'`
LANG=`echo $new_language|sed -e 's,.* ,,'`

#LANG=`echo $new_x_server_defs | sed -e "s, .*,,"`
[ "$LANG" ] || { LANG="de"; echo -e "$0: No language \
variable set for keyboard layout." >>$LOGFILE; }
sed -e "s,KEYTABLE=.*,KEYTABLE=\"$KEYTABLE\"," \
-e "s,RC_LANG=.*,RC_LANG=\"$LANG\"," \
-e "s,START_GPM=.*,START_GPM=$start_gpm," \
-e "s,GPM_.*,GPM_PARAM=\"-t $MP -m $MD\"," \
-e "s,START_SNMP.*,START_SNMPD=$new_start_snmp,"\
-e "s,START_SSHD.*,START_SSHD=$new_start_sshd," \
-e "s,DISPLAYM.*,DISPLAYMANAGER=$new_start_xdmcp," \
-e "s,DEFAULT_WM.*,DEFAULT_WM=\"$defaultwm\"," \
-e "s,CRON.*,CRON=\"$new_start_cron\"," \
-e "s,START_RWHOD.*,START_RWHOD=$new_start_rwhod," \
-e "s,START_LPD.*,START_LPD=\"$start_lpd\"," \
-e "s,START_CUPS.*,START_CUPS=\"$start_cups\"," \
-e "s,START_YPBIND.*,START_YPBIND=\"$start_nis\"," \
-e "s,START_XNTPD.*,START_XNTPD=\"$start_xntp\"," \
-e "s,XNTPD_I.*,XNTPD_INITIAL_NTPDATE=\"$initntp\"," \
-e "s,START_X=.*,START_X=\"$new_start_x\"," \
-e "s,VMWARE.*,VMWARE=\"$start_vmkernel\"," \
-e "s,VMIMAGE.*,VMIMAGE=\"$new_vmware\"," \
-e "s,DEBUGLEVEL.*,DEBUGLEVEL=\"$DXSDEBUG\"," \
/etc/rc.config.default | grep -v "#" >/etc/rc.config;
```

```
# set up cron if started
test "$new_start_cron" = yes && \
    { cat /etc/crontab.default > /etc/crontab
    echo -e `echo -e "$new_crontab_entries"` >> /etc/crontab; }

exit 0
```

Die Einrichtung von Bild und Ton: hwsetup Dieses Skript kennt keine Kommandozeilenoptionen. Es wird beim Start des Diskless Clients zu einem frühen Zeitpunkt aus der */etc(.s)/init.d/boot* heraus (in den Hintergrund, im Debuglevel "2" im Vordergrund) aufgerufen. Alle statischen Festlegungen werden zu Beginn der Datei getroffen. Hierzu zählen z.B. die verschiedenen Modlines für die spätere *XF86Config*. Anpassungen sind für TFT-Displays und spezielle Röhrenmonitore notwendig. Weiterhin können eine Reihe von Default-Werten definiert werden, wenn keine Daten per DHCP (aus */var/state/dhcp/dhclient.leases*) oder **hwinfo**[27] erhältlich sind. Sollte sich das Laden bestimmter Module als problematisch herausstellen, kann die Reihenfolge innerhalb der **hwsetup** angepasst werden.

```
#!/bin/sh
#
# hwsetup for hardware checking and configuration for linux diskless
# clients, sets up /etc/X11/XF86Config, /etc/fstab, /etc/auto.misc
#
# a lot of configuration is done via the pre-init script and dhclient-
# script (located in /var/ram/sbin and used within the inittar), consult
# these scripts too ...
#
# DEBUGLEVEL greater than one produce a lot of additional information,
# you can find it in the logfile (/var/log/hwsetup.log)
#
version="0.0.6a"
#
# Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
################################################################################

# policies for hardware detection:
#
# mouse -> dhcp,hwinfo,default
# graphics -> dhcp(checked with hwinfo),hwinfo,default
# monitor -> dhcp,hwinfo,default

# check for audio devices and load dummy driver if no can be found

# check for floppy/cdrom/dvd devices and add them to /etc/fstab

# internal functions
# writing the error/info log (consult this if you encounter problems
# configuring your hardware)
logwrite () {
    echo -e "hwsetup: $1" >> $LOGFILE
}
# check for glx capabilities of the graphic adaptor and try to load the
# soft glx driver if no other is available
glx_check () {
    [ $DEBUGLEVEL -gt 1 ] && logwrite "checking for the existence of glx \
library files. they should be installed\nbecause a lot of todays software \
makes use of it."
    rm /RAM/etc/X11/modules/libglx.a >> $LOGFILE 2>&1
    LIBGLX=`ls /usr/X11R6/lib/modules/extensions/libglx.a 2>/dev/null` || \
    LIBGLX=`ls /usr/X11R6/lib/modules/extensions/libglx.a*xf86_glx* 2>/dev/null`
    LIBGLC=`ls /usr/X11R6/lib/modules/extensions/libGLcore.a 2>/dev/null` || \
    LIBGLC=`ls /usr/X11R6/lib/modules/extensions/libGLcore.a*xf86_glx* \
2>/dev/null`
    if ([ "$LIBGLX" ] && [ "$LIBGLC" ]) ; then
```

[27] Dieses Tool ist SuSE-spezifisch, Anpassungen an die automatische Hardwareerkennung anderer Distributionen existieren bisher nicht.

```
ln -s $LIBGLX /RAM/etc/X11/modules/libglx.a >> $LOGFILE 2>&1
ln -s $LIBGLC /RAM/etc/X11/modules/libGLcore.a >> $LOGFILE 2>&1
return 0
    else
return 1
    fi
}

##############################################################################
# predifine some variables for XF86Config (if no dhcp variable or auto
# detected info is available)
XF86CONFFILE="/RAM/etc/X11/XF86Config"
HSYNCRANGE="31.5-63.5"
VSYNCRANGE="60-90"
DEFAULTLANG="de"
DEFAULTCOLORDPT=16

# get some additional variables (i.e. DEBUGLEVEL)
. /etc/rc.config

# set variables representing the XF86Config sections (version 4.X)
Module='\tLoad\t\t"dbe"\n
\tLoad\t\t"extmod"\n
\tLoad\t\t"type1"\n
\tLoad\t\t"speedo"\n
\tLoad\t\t"freetype"\n
\tLoad\t\t"v4l"'

ServerFlags='\tOption\t\t"AllowMouseOpenFail"'

Files='\tRgbPath\t\t"/usr/X11R6/lib/X11/rgb"\n
\tModulePath\t"/RAM/etc/X11/modules"\n
\tModulePath\t"/usr/X11R6/lib/modules"\n
\tFontPath\t"/usr/X11R6/lib/X11/fonts/misc/:unscaled"\n
\tFontPath\t"/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"\n
\tFontPath\t"/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"'

InputDevice='\tIdentifier\t"Keyboard1"\n
\tDriver\t\t"keyboard"\n
\tOption\t\t"XkbRules"      "xfree86"\n
\tOption\t\t"XkbLayout"     "LANG"\n
\tOption\t\t"XkbModel"      "pc102"'

InputMouse='\tIdentifier  "Mouse1"\n
\tDriver      "mouse"\n
\tOption      "Protocol"      "MP"\n
\tOption      "Device"        "/dev/MD"\n
\tOption      "Emulate3Buttons"\n
\tOption      "ZAxisMapping"  "4 5"\n
\tOption      "Buttons"       "3"'

Monitor='\tIdentifier "Default"\n
\tOption\t"CalcAlgorithm" "CheckDesktopGeometry"\n
\tHorizSync\tHS\n
\tVertRefresh\tVS\n
\tUseModes\t"Default"\n
\tOption\t"DPMS"'

Modes='\tIdentifier "Default"'

Modelines='
\t# crt modelines (refreshrates should be above 72Hz or enable uncommented lines)\n
\tModeline "640x400"    25.175 640  664  760  800  400  409  411  450\n
\tModeline "640x400"    31.5   640  672 736   832  400  401  404  445\n
\t#Modeline "640x480"     31.50   640  680  720  864   480  488  491  521\n
\tModeline "640x480"     45.80   640  672  768  864   480  488  494  530\n
\t#Modeline "800x600"     50.00   800  856  976 1040   600  637  643  666\n
\tModeline "800x600"     69.65   800  864  928 1088   600  604  610  640\n
\t#Modeline "1024x768"    44.90  1024 1048 1208 1264   768  776  784  817 interlace\n
\t#Modeline "1024x768"    80.00  1024 1052 1164 1360   768  784  787  823\n
\tModeline "1024x768"    86.00  1024 1040 1152 1360   768  784  787  823\n
```

```
\tModeline "1024x768"    98.90  1024 1056 1216 1408   768  782  788  822\n
\tModeline "1024x768"   115.50  1024 1056 1248 1440   768  771  781  802\n
\tModeline "1152x864"    92.00  1152 1208 1368 1474   864  865  875  895\n
\tModeline "1152x864"   110.00  1152 1240 1324 1552   864  864  876  908\n
\t#Modeline "1280x960"   112.00  1280 1312 1456 1704   960  963  970 1064\n
\tModeline "1280x960"   142.00  1280 1312 1456 1712   960  963  970 1064\n
\t#Modeline "1280x1024"  145.00  1280 1312 1456 1712  1024 1027 1030 1064\n
\tModeline "1280x1024"  157.50  1280 1344 1504 1728  1024 1025 1028 1072\n
\tModeline "1400x1050" 180.00 1400 1472 1672 1880  1050 1052 1055 1100\n
\tModeline "1600x1200"  202.50  1600 1664 1856 2160  1200 1201 1204 1250\n
\t# tft modlines (refreshrates of 60Hz schould be ok)\n
\tModeline "lcd1024x768"    60.00  1024 1052 1164 1360   768  784  787  823\n
\tModeline "lcd1280x1024"  108.00  1280 1328 1440 1688  1024 1025 1028 1066\n
\tModeline "lcd1400x1050"  160.00  1400 1472 1672 1880  1050 1052 1055 1100'

Device='\tIdentifier\t"StdGraphics"\n
\tDriver\t\t"DRV"'

Screen='\tIdentifier\t"Screen 1"\n
\tDevice\t"StdGraphics"\n
\tMonitor\t"Default"\n
\tDefaultColorDepth CDP'

ServerLayout='\tIdentifier\t"Simple Layout"\n
\tScreen\t"Screen 1"\n
\tInputDevice "Mouse1"\t"CorePointer"\n
\tInputDevice "Keyboard1"\t"CoreKeyboard"'

DRI='\tGroup\t\t"video"\n
\tMode\t\t0666'

# end of predefinitions
#
###############################################################################
#
# preparations

# disable the writing of a logfile if loglevel equals to zero, enable much
# kernel level debugging output if loglevel is higher then one
PRINTK=`cat /proc/sys/kernel/printk`
echo "0 0 0 0" >/proc/sys/kernel/printk
if [ "$DEBUGLEVEL" != "0" ]; then
        LOGFILE="/var/log/hwsetup.log"
[ $DEBUGLEVEL -gt 1 ] && { logwrite "\n\t!! not all output produced \
within $0 makes sense in all means.\n\t!! But you may find some useful \
information about module loading and\n\t!! errors produces within this \
context. The kernel message output will\n\t!! be rised to a very high \
level.\n\n";
echo "7 7 7 7" >/proc/sys/kernel/printk; }
else
        LOGFILE="/dev/null"
fi

logwrite "Running version $version"

# get X server definitions evetually gotten via dhclient process
#XSRVDEF=( `grep x-server-defs /var/lib/dhcp/dhclient.leases |\
# awk -F \" {'print $2'}` )
#LANG=${XSRVDEF[0]}
#MP=${XSRVDEF[1]}
#MD="/dev/"${XSRVDEF[2]}
#HS="31.5-"${XSRVDEF[3]}
#VS="50-"${XSRVDEF[4]}
#MAXRES=${XSRVDEF[5]}
#DRV=${XSRVDEF[6]}
#CDP=${XSRVDEF[7]}
LANG=( `grep language /var/lib/dhcp/dhclient.leases |\
awk -F \" {'print $2'}` )
MOUSEDEF=( `grep hw-mouse /var/lib/dhcp/dhclient.leases |\
awk -F \" {'print $2'}` )
MP=${MOUSEDEF[0]}
```

```
MD="/dev/"${MOUSEDEF[1]}
MONITORDEF=( `grep hw-monitor /var/lib/dhcp/dhclient.leases |\
awk -F \" {'print $2'}` )
HS=`echo ${MONITORDEF[0]}| sed -e 's,k[hHz]*,,'`
VS=`echo ${MONITORDEF[1]}| sed -e 's,[hH]z*,,'`
MAXRES=${MONITORDEF[2]}
GRAPHICDEF=( `grep hw-graphic /var/lib/dhcp/dhclient.leases |\
awk -F \" {'print $2'}` )
DRV=${GRAPHICDEF[0]}
CDP=${GRAPHICDEF[1]}

# check for hwinfo tool (SuSE specific)
# in later versions some other hardware detection techniques should be
# supported too ...
if HWINFO=`which hwinfo` ; then
test -x $HWINFO || { HWINFO="no" ;
logwrite "Tool 'hwinfo' is not executable."; }
else
HWINFO="no"
logwrite "Tool 'hwinfo' could not be found."
fi


# load modules (needed eventually for auto detection of harware components)
[ $DEBUGLEVEL -gt 1 ] && logwrite "loading several kernel modules needed \
for further autoprobing of\nhardware. The modules for ie1394 are tried \
independently of existence of\nany apropriate device."
modprobe -aq usbcore serial >> $LOGFILE 2>&1
modprobe -q agpgart >> $LOGFILE 2>&1 || \
  { modprobe -q agpgart agp_try_unsupported=1 >> $LOGFILE 2>&1;
logwrite "problems loading agpgart support -> trying 'unsupported'"; } || \
logwrite "no agpgart -> no dri/glx available then";

# end of preparations
#
###############################################################################
#
# set up usb subsystem

[ $DEBUGLEVEL -gt 1 ] && logwrite "starting with usb setup. it will be needed \
at least if a usb mouse should be\nused with XFree86."
if [ -d /proc/bus/usb ]; then
# if it's not mounted, try to mount it
        if [ ! -f /proc/bus/usb/devices ]; then
mount /proc/bus/usb || logwrite "failed to mount /proc/bus/usb"
fi
fi
for i in usb-uhci uhci usb-ohci ehci-hcd ohci-hcd uhci-hcd; do
modprobe -q $i >> $LOGFILE 2>&1 && {
logwrite "loaded HCD: $i"; USB=yes; }
done
# loading coldplugged mouse for later configuration for XFree86
# sleep may be needed for registering of plugged usb devices
( [ $MD = "/dev/input/mice" ] && { modprobe -a hid input mousedev \
>> $LOGFILE 2>&1 || logwrite \
"failed to load usb/mouse modules for coldplugged mouse"; } ) &

# set up hotplugging
# -> should be done sometime ...
[ $DEBUGLEVEL -gt 1 ] && logwrite "hotplugging is not fully functional ... \
some work/understanding of concepts\nis to be done."

# loading modules for later periphery (ide and ieee1394) detection
modprobe -aq floppy ide-probe-mod ide-mod pcilynx ohci1394 >> $LOGFILE 2>&1 &

# write info to log file
logwrite "Setting up XF86Config ->\nThe following variables are set:\
\nLanguage: $LANG\nMouseProtocol: $MP\nMouseDevice: $MD\n\
HorizontalSync: $HS\nVerticalRefr: $VS\nMaxResolution: $MAXRES\n\
XFreeDriver: $DRV\nDefColorDepth: $CDP"
```

```
# do some checks on variables and commands needed for XF86Config setup
[ -z $LANG ] && LANG=$DEFAULTLANG
if [ -z $CDP ] ; then CDP=$DEFAULTCOLORDPT
else
[ $CDP != "8" ] && [ $CDP != "15" ] && [ $CDP != "16" ] && \
[ $CDP != "24" ] && [ $CDP != "32" ] && { CDP=$DEFAULTCOLORDPT; \
logwrite "Colordepth variable (set to $CDP at the moment) \
should be\n8,15,16,24 or 32 bpp."; }
fi
[ -x /bin/sed ] || \
logwrite "'sed' not found or not executable. It is used to \
create the XF86Config file."
[ -x /bin/grep ] || \
logwrite "'grep' not found or not executable. It is used to \
create the XF86Config file."

# trying to identify the mouse sleep may be needed for detection of usb
# mice
[ $MD = "/dev/" ] && [ -z "$MP" ] && [ x$HWINFO != "xno" ] && {
[ $USB ] && sleep 2
hwinfo --mouse >/tmp/hwsetup.tmp
MOUSE=`cat /tmp/hwsetup.tmp | grep -e "File:" -e XFree86`
[ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of 'hwinfo --mouse':\n" ;
cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }
ADMD=`echo $MOUSE | sed -e "s/.*e: //g" -e "s/ XFree.*//g"`
ADMP=`echo $MOUSE | sed -e "s/.*l: //g"`
logwrite "got mousedev: $ADMD, mouseprotocol: $ADMP from autodetection"
[ $ADMD ] && { MD=$ADMD; logwrite "Using $ADMD as mouse device \
gotten via 'hwinfo'"; }
[ $ADMP ] && { MP=$ADMP; logwrite "Using $ADMP as mouse protocol \
gotten via 'hwinfo'"; }
[ $ADMD = "/dev/input/mice" ] && modprobe -a hid input mousedev \
>> $LOGFILE 2>&1 || logwrite \
"failed to load usb/mouse modules for detected mouse"
}
# if mouse couldn't be detected set defaults
[ $MD = "/dev/" ] && MD="/dev/psaux"
[ -z $MP ] && MP="PS/2"

# unload serial driver if not needed for mouse operation
expr $MD : .*ttyS.* >> $LOGFILE 2>&1 || rmmod serial >> $LOGFILE 2>&1

# trying to identify the graphics adaptor
[ x$HWINFO != "xno" ] && hwinfo --display >/tmp/hwsetup.tmp
[ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of 'hwinfo --display':\n" ;
cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }

#
for i in `cat /tmp/hwsetup.tmp | grep v4 | sed -e "s,.*: ,,g"`
do HWIDRV=$i; break
done

[ -z $DRV ] && DRV="vesa"
if [ $HWIDRV ] ; then
if [ $DRV != $HWIDRV ] ; then DRV=$HWIDRV; fi
fi

# configure software/hardware opengl extension
GLX="glx"
rm /RAM/etc/X11/modules/libglx.a >> $LOGFILE 2>&1
if [ $DRV = "nv" ] || [ $DRV = "nvidia" ] ; then
    KVER=`uname -r 2>/dev/null`
    KMOD=`ls /lib/modules/$KVER/kernel/drivers/video/nvidia.o 2>/dev/null`
    LIBGLX=`ls /usr/X11R6/lib/modules/extensions/libglx.so 2>/dev/null` || \
    LIBGLX=`ls /usr/X11R6/lib/modules/extensions/libglx.so*nv* \
2>/dev/null`
    if ([ -n $KMOD ] && [ -n $LIBGLX ]); then
GLXLIBVER=`strings $LIBGLX|grep " id:"|awk '{print$7}'`
GLXKERNEL=`strings $KMOD|grep " id:"|awk '{print$9}'`
if [ "$GLXKERNEL" = "$GLXLIBVER" ] ; then
    rmmod agpgart >> $LOGFILE 2>&1
```

```
    if insmod nvidia >> $LOGFILE 2>&1 ; then
DRV=nvidia
if [ "$LIBGLX" ] ; then
    ln -s $LIBGLX /RAM/etc/X11/modules/libglx.a >> $LOGFILE 2>&1
    [ $DEBUGLEVEL -gt 1 ] && logwrite "All nvidia related \
stuff seems to be configured. if Xfree86 isn't working\nproperly there may \
be some hardware incompatibilities left... Remove nvidia\n kernel module then \
or modify the hwsetup script."
    logwrite "Check if problems occur that libglx.so nvidia \
XFree86 module is of the same version as the library used!";
else
    GLX=no ;
    logwrite "Check libglx.so nvidia XFree86 module! Unable to \
properly link this file (not existing/several versions etc.) ...";
    glx_check && GLX=glx; DRV=nv;
fi
    else
logwrite "Failed to load detected nvidia.o kernel module.";
modprobe agpgart >> $LOGFILE 2>&1
    fi
    else DRV=nv ;

glx_check || \
{ GLX=no ; logwrite "Unable to locate XFree86 libglx module"; }
    fi
    fi
else
glx_check || { GLX=no ;
logwrite "No glx extension libraries found. Disabling glx."; }
fi
# check for dri extension
[ x$HWINFO != "xno" ] && cat /tmp/hwsetup.tmp | grep " dri" >/dev/null \
&& { modprobe -qa $DRV >> $LOGFILE 2>&1; } && DRM=yes

# try to gather monitor data, take given values or set save defaults
[ x$HWINFO != "xno" ] && { hwinfo --monitor >/tmp/hwsetup.tmp
[ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of 'hwinfo --monitor':\n" ;
cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }
MONITOR=`cat /tmp/hwsetup.tmp | grep -e Vert -e Hor`
if [ "$MONITOR" ] ; then
VS=`echo $MONITOR | sed -e "s,Vert. Sync Range: ,,i" -e "s, Hz.*,,i"`
HS=`echo $MONITOR | sed -e "s,.*Hor. Sync Range: ,,i" -e "s, kHz.*,,i"`
else
[ x$HS = "x" ] && HS=$HSYNCRANGE
[ x$VS = "x" ] && VS=$VSYNCRANGE
fi; }

# compute max resolution
for MR in 640x400 640x480 800x600 1024x768 1280x1024 1400x1050 1600x1200
do MODES="\"$MR\" \"lcd$MR\" $MODES"
if [ $MR = "$MAXRES" ] ; then break ; fi
done


# write XF86Config file
echo -en "#\n# autogenerated XF86Config by hwsetup\n# \
Dirk von Suchodoletz <dirk@goe.net>, " >$XF86CONFFILE || \
logwrite "unable to create the XF86Config file."
date >>$XF86CONFFILE
echo -e "# DO NOT EDIT THIS FILE BUT '$0' INSTEAD!\n#" \
>>$XF86CONFFILE
for section in Files ServerFlags Module InputDevice Monitor \
    Modes Screen Device ServerLayout DRI ; do
    echo "Section \"$section\"" >>$XF86CONFFILE
    echo -e ${!section} | sed -e "s,LANG,$LANG,g" -e "s,HS,$HS,g"    \
 -e "s,VS,$VS,g" -e "s,MODES,$MODES,g" -e "s,MODES,$MODES,g" \
 -e "s,DRV,$DRV,g" -e "s,CDP,$CDP,g" >>$XF86CONFFILE
    case "$section" in
Files)
    for i in /usr/X11R6/lib/X11/fonts/* ; do
    echo -e "\tFontPath\t\"$i/\"" >>$XF86CONFFILE ; done
```

```
;;
Device)
    # special options may be needed for some drivers
    [ $DEBUGLEVEL -gt 1 ] && logwrite "adding for some Xfree86 modules \
(radeon, s3virge, nvidia) special options to\nthe 'Device' section. Consult \
hwsetup if you like to add more or remove some."
    [ x$DRV = "xs3virge" ] && {
    echo -e "\tOption\t\t\"XVideo\" \"Off\"\n" >>$XF86CONFFILE
    [ $DEBUGLEVEL -gt 1 ] && logwrite "added special option for \
the s3virge module -> 'XVideo Off'. remove the\nappropriate line in hwsetup \
if you like not to use it." ; }
    [ x$DRV = "xnvidia" ] && {
    echo -e "\tOption\t\t\"NvAGP\" \"1\"\n" >>$XF86CONFFILE
    [ $DEBUGLEVEL -gt 1 ] && logwrite "added special option for \
the nvidia module -> 'NvAGP '. it sets the use\nof internal or external \
agpgart support." ; }
    [ x$DRV = "xradeon" ] && {
    echo -e "\tOption\t\t\"BusID\" \"1:0:0\"\n" >>$XF86CONFFILE
    [ $DEBUGLEVEL -gt 1 ] && logwrite "added special option for \
the radeon module -> 'BusID'. it defines the\npostion of graphics adaptor \
in pci/agp bus needed for acceleration." ; }
;;
Module)
    [ x$DRM = "xyes" ] && \
    echo -e "\tLoad\t\t\"dri\"" >>$XF86CONFFILE
    [ x$GLX != "xno" ] && \
    echo -e "\tLoad\t\t\"$GLX\"" >>$XF86CONFFILE
;;
InputDevice)
    [ $LANG = "de" ] && \
    echo -e '\tOption\t\t"XkbVariant"    "nodeadkeys"' >>$XF86CONFFILE
    echo -e "EndSection\nSection \"$section\"" >>$XF86CONFFILE
    echo -e $InputMouse | sed -e "s,MP,$MP,g"  \
-e "s,/dev/MD,$MD,g" >>$XF86CONFFILE
;;
Modes)
    for MR in 640x400 640x480 800x600 1024x768 1280x1024 1400x1050 1600x1200
do echo -e $Modelines | grep -e "lcd"$MR -e $MR >>$XF86CONFFILE
if [ $MR = "$MAXRES" ] ; then break ; fi
    done
;;
Screen)
    for BPP in 8 15 16 24 32
do echo -e '\tSubSection "Display"\n\t\tDepth\t'\
$BPP'\n\t\tModes\t'$MODES'\n\tEndSubSection' >>$XF86CONFFILE
if [ $BPP = "$CDP" ] ; then break ; fi
    done
         ;;
esac
echo -e "EndSection\n" >>$XF86CONFFILE
done

# produce lower resolution XF86Config
if [ $MR=1280x1024 ] || [ $MR=1400x1050 ] || [ $MR=1600x1200 ] ; then \
    sed -e "s,Modes[ ]*\"$MAXRES\" \"lcd$MAXRES\",Modes       ,g" \
$XF86CONFFILE >$XF86CONFFILE.lowres
fi
# linking the XFree86 executable (use other techniques for XFree86 version 3)
ln -sf /usr/X11R6/bin/XFree86 /RAM/run/X >> $LOGFILE 2>&1 || \
logwrite "'ln' command not found or failed"

# preparing inittab for enabling graphical user interface
sed -e "s,#7:,7:,g" /etc/inittab >/etc/inittab.new
mv /etc/inittab.new /etc/inittab
telinit q

# setting up sound module
[ x$HWINFO != "xno" ] && {
    # oss and kernel modules named with activation command
    hwinfo --sound  >/tmp/hwsetup.tmp
    [ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of 'hwinfo --sound':\n" ;
```

```
cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }
    cmd=‘cat /tmp/hwsetup.tmp | grep "Cmd:" | awk -F \" {’print $2’}‘
    if [ -n "$cmd" ] ; then $cmd
    # alsa seem to be named by info line only, if no adaptor is detected use
    # the dummy module; load oss compatibility modules afterwards
    else
modules=‘cat /tmp/hwsetup.tmp | grep "Info:" | \
awk -F : {’print $2’} | sed -e "s/,/ /"‘
if [ -n "$modules" ] ; then
    modprobe -a $modules >> $LOGFILE 2>&1
    [ $DEBUGLEVEL -gt 1 ] && logwrite "if no module could be loaded \
at this stage, check the names, the existence\nof the modules or \
configuration within modules.conf."
else
    logwrite "Unable to find an audio device, using snd-dummy.";
    [ $DEBUGLEVEL -gt 1 ] && logwrite "check the name of your \
soundcard with ’lspci’ or similar, the type\nof chip used with it and if \
it is supported by some driver."
    modprobe snd-dummy >> $LOGFILE 2>&1;
fi
    modprobe -a snd-mixer-oss snd-pcm-oss >> $LOGFILE 2>&1
    fi
}

# setting up floppy and cdrom/dvd device(s)
[ x$HWINFO != "xno" ] && { hwinfo --floppy >/tmp/hwsetup.tmp ;
[ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of ’hwinfo --floppy’:\n" ;
    cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }
for device in ‘cat /tmp/hwsetup.tmp | grep "Device File" | \
sed -e "s,.*: ,,g"‘; do
    case $device in
/dev/hd*|/dev/sd*)
    modprobe ide-floppy >> $LOGFILE 2>&1
    echo -e "${device}4\t\t/mnt/zip\tauto\t noauto,rw,users 0  0" \
>>/RAM/etc/fstab
    echo -e "${device}4\t\t/misc/zip\tauto\t noauto,rw,users 0  0" \
>>/RAM/etc/fstab
    echo -e "zip\t\t-fstype=auto,rw,umask=000 :${device}4" \
>>/RAM/etc/auto.misc
    echo -e "zip\t\t-fstype=auto,rw,umask=000 :${device}1" \
>>/RAM/etc/auto.misc
;;
/dev/fd*)
    echo -e "${device}\t\t/mnt/floppy\tauto\t noauto,rw,users 0  0" \
>>/RAM/etc/fstab
    echo -e "${device}\t\t/misc/floppy\tauto\t noauto,rw,users 0  0" \
>>/RAM/etc/fstab
    echo -e "floppy\t\t-fstype=auto,rw,umask=000\t :/dev/fd0" \
>>/RAM/etc/auto.misc
;;
    esac
done
cat /tmp/hwsetup.tmp | grep "/dev/fd" >> $LOGFILE 2>&1 || \
rmmod floppy >> $LOGFILE 2>&1 ; }

# setting up cdrom/dvd device(s)
[ x$HWINFO != "xno" ] && { hwinfo --cdrom >/tmp/hwsetup.tmp ;
[ $DEBUGLEVEL -gt 1 ] && { logwrite "---> output of ’hwinfo --cdrom’:\n" ;
    cat /tmp/hwsetup.tmp >> $LOGFILE 2>&1 ; }
unset i
for device in ‘cat /tmp/hwsetup.tmp | grep "Device File" | sed -e "s,.*: ,,g"‘
do
    echo -e "${device}\t\t/mnt/cdrom$i\tauto\t noauto,rw,users 0  0
${device}\t\t/misc/cdrom$i\tauto\t noauto,rw,users 0  0" >>/RAM/etc/fstab
    echo -e "cdrom$i\t\t-fstype=iso9660,norock,umask=000 :$device" \
>>/RAM/etc/auto.misc
    [ $i ] || { ln -sf $device /dev/cdrom >> $LOGFILE 2>&1 ;
echo -e "/dev/cdrom\t\t/mnt/cdrom$i\tauto\t noauto,rw,users 0  0" \
>>/RAM/etc/fstab
modprobe -a ide-cd cdrom >> $LOGFILE 2>&1
#echo 1 >/proc/sys/dev/cdrom/autoeject
```

```
# write out cdrom/dvd drive info if in higher DEBUGLEVEL
[ $DEBUGLEVEL -gt 1 ] && cat /proc/sys/dev/cdrom/info >> $LOGFILE 2>&1
    }
    i=1
done
# remove unused modules
cat /tmp/hwsetup.tmp | grep "/dev/hd*" >> $LOGFILE 2>&1 || \
rmmod cdrom ide-mod ide-probe-mod >> $LOGFILE 2>&1;
rm /tmp/hwsetup.tmp >> $LOGFILE 2>&1; }

# restore kernel messaging status to normal if DEBUGLEVEL is one
[ $DEBUGLEVEL = 1 ] && { echo "$PRINTK" >/proc/sys/kernel/printk;
    logwrite "normalising kernel messaging status to $PRINTK.\n"; }
logwrite "hardware setup done.\n\n"
exit 0
```

Die Auswahl des Betriebsmodus der grafischen Oberfläche erfolgt mittels **startgui**. Dieses Skript wird üblicherweise von **init**[28] aufgerufen, nach den Einträgen in der */etc(.s)/ inittab*. Es nimmt als Kommandozeilenoption den Window-Manager an, wobei durchaus auch der Citrix-Metaframe-Client und VMware als Windowmanager gesehen werden. Aus diesem Skript kann wiederum **runvmware** mit einem bestimmten Image als Parameter gestartet werden.

```
#!/bin/sh
#
# /sbin/startgui (runs as script replacement for xinit)
#
# version 0.0.2a
#
# Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $
#
##############################################################################

# functions
error () {
$2 $1
}

sysresources=/usr/X11R6/lib/X11/Xresources
sysmodmap=/usr/X11R6/lib/X11/Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

#if [ -f $sysmodmap ]; then
#    xmodmap $sysmodmap
#fi

#if [ -f $userresources ]; then
#    xrdb -merge $userresources
#fi

#if [ -f $usermodmap ]; then
#    xmodmap $usermodmap
#fi
WM=$1
[ $WM ] || WM=kde
case "$WM" in
*fvwm*)
     fvwm2
;;
```

[28] Dem "richtigen" **init** des konfigurierten Clients, nicht vom ersten Skript innerhalb des INITTAR.

```
*kde*)
mkdir ~/.kde
cp -a /etc.s/dxs/WMsettings/kde ~/.kde
kde
;;
*wmaker*|*windowmaker*)
mkdir ~/GNUstep
cp -a /etc.s/dxs/WMsettings/windowmaker ~/GNUstep
wmaker
;;
*ice*)
/usr/X11R6/bin/icewm
;;
*gnome*)
cp -a /etc.s/dxs/WMsettings/gnome/.* ~/
/usr/X11R6/bin/gnome
;;
*citrix|Citrix|ICAClient|*wfica)
xsetroot -solid black
cp -a /etc.s/dxs/ICAClient/ ~/.ICAClient
/usr/lib/ICAClient/wfica
;;
*citrix-mgr|*citrix-indirect|*wfcmgr)
xsetroot -solid black
cp -a /etc.s/dxs/ICAClient/ ~/.ICAClient
#
# indirect bauen
#
;;
*vmware*)
. /etc/rc.config
xsetroot -solid black
/bin/runvmware $VMIMAGE
;;
*)
error "kein sinnvoller wm angegeb." xmessage
kde
;;
esac
```

Die Konfiguration von VMware geschieht mittels **runvmware**. Dieses Skript legt eine gültige VMware-Konfiguration *autovm.conf* im Verzeichnis *.vmware* an, ohne dabei beim Start auf das eventuelle Vorhandensein der Datei zu prüfen. Das Skript übernimmt in der derzeitigen Version als Kommandozeilenparameter den Namen der Imagedatei, wie sie unterhalb von */usr/share/vmware* erwartet wird. Einige Variable, wie die MAC-Adresse, Rechner- und Username, werden automatisch ermittelt, andere per Zuweisung festgelegt. Weiterhin übeprüft das Skript auf das Vorhandensein bestimmter Laufwerke, wie CD-Rom oder Diskette. Mit diesen Daten wird die Konfigurationsdatei bestückt und die Registry-Dateien erzeugt.

```
#!/bin/sh
#
# Author: Dirk von Suchodoletz <dirk@goe.net>, 26-03-2003
#
# script for preparing VMware environment on linux diskless clients
# (local version)

# check for apic enabled!
# cat /proc/cmdline |grep apic &>/dev/null && OK

# create vmware config directory
mkdir ~/.vmware &>/dev/null

# interpreting command line options
(( NARG=0 ))
TEST=( $@ )
while [ $NARG -le $# ] ; do
```

```
        case "${TEST[$NARG]}" in
        *help*|-h)
                usage
                exit 0
        ;;
win98*)
FIRSTIDE="/usr/share/vmware/win98"
OSTYPE="win98"
DISPLAYNAME="Windows 98"
#ln -fs /usr/share/vmware/"${TEST[$NARG]}" ~/.vmware/w98 &>/dev/null
;;
winxphome*|win_xp_home*|WinXP_Home*|winXPHome*)
FIRSTIDE="/usr/share/vmware/win98"
OSTYPE="winXPHome"
DISPLAYNAME="Windows XP Home"
;;
        esac
(( NARG=$NARG+1 ))
done

# remove leftover locks from former runs, problems may occure if you run
# more then one vmware with this script
rm -rf ~/.vmware/*LOCK &>/dev/null

# configuring MAC address: first four bytes are fixed (00:50:56:0D) the
# last two bytes are taken from the local network adaptor
MAC=‘/sbin/ifconfig eth0|grep HWaddr|awk ’{print$5}’|awk -F : \
’{print$5":"$6}’‘

# gather memory info and set virtual machine memory size accordingly
declare -i MEM
MEM=‘cat /proc/meminfo|grep MemTotal|sed -e ’s,[0-9][0-9][0-9] kB,,’ -e ’s,MemTotal: *,,’‘
if [ $MEM -le 96 ] ; then
MEM=0
elif [ $MEM -le 128 ] ; then
MEM=$MEM-50
elif [ $MEM -le 160 ] ; then
MEM=$MEM-64
elif [ $MEM -le 288 ] ; then
MEM=$MEM-96
elif [ $MEM -gt 288 ] ; then
MEM=$MEM-128
elif [ $MEM -gt 500 ] ; then
MEM=$MEM-192
fi
# to get 4MByte boundary
MEM=$MEM/4*4

# set samba workgroup
WORKGROUP="tux-net"

# set hostname: using original hostname and adding string "-vm"
HOSTNAME=‘hostname‘"-vm"

# place for the virtual floppy "B:"
FILENAME="/tmp/scratch/"‘hostname‘"-1440.img"
COMMENT="Diskless Workstation VMware-Image"

# OS to run (titleline, type of os, ... in autovm.conf)

# look for cdrom and floppy and add them to the vm config file
if [ -L /dev/cdrom ] ; then
    CDR=TRUE
else
    CDR=FALSE
fi
if [ -L /dev/fd0 ] ; then
    FLOPPYA=TRUE
else
    FLOPPYA=FALSE
fi
```

```
# create vmware subdirectory within the users home and copy configuration
# files
mkdir ~/.vmware &>/dev/null
cp /etc.s/dxs/VMware/run.bat /mnt/loop0
cp /etc.s/dxs/VMware/license.ws.3.0 /etc.s/dxs/VMware/preferences \
/etc.s/dxs/VMware/nvram ~/.vmware
sed -e "s,00:00\",$MAC\",i" -e "s,FLOPPY-B,$FILENAME,i" \
    -e "s,ide0:1\.present =.*,ide0:1\.present = \"$CDR\"," \
    -e "s,floppy0\.startConnected =.*,floppy0\.startConnected = \"$FLOPPYA\"," \
-e "s,memsize =.*,memsize = \"$MEM\"," \
-e "s,ide0:0.fileName = .*,ide0:0.fileName = \"$FIRSTIDE\"," \
-e "s,displayName = .*,displayName = \"$DISPLAYNAME\"," \
-e "s,guestOS = .*,guestOS = \"$OSTYPE\"," \
/etc.s/dxs/VMware/vmware.conf.default >~/.vmware/autovm.conf

#   -e "s,floppy0\.present =.*,floppy0\.present = \"$FLOPPYA\"," \

# prepare the registry files for the client windows os
sed -e "s,\"WG\",\"$WORKGROUP\",i" -e "s,\"CN\",\"$HOSTNAME\",i" \
-e "s,\"CMT\",\"$COMMENT\",i" /etc.s/dxs/VMware/rechner.reg >\
/mnt/loop0/rechner.reg
for i in logon.reg home.reg; do
    sed -e "s,USER\",$USER\",g" \
/etc.s/dxs/VMware/$i >/mnt/loop0/$i
done

# set the appropriate permissions for the vmware config file
chmod u+rwx ~/.vmware/autovm.conf

# later dependent on OS to run
#ln -s /misc/win/w98 ~/.vmware/w98 &>/dev/null

# run vmware with the automatically written config file
vmware ~/.vmware/autovm.conf -- -geometry +51+50
```

C Eine kurze Installationsanleitung

C.1 Unterstützte Linux-Distributionen

Das hier beschriebene Projekt setzt direkt auf der SuSE-Linux-Distribution auf und bezieht bestimmte Programme, wie **hwinfo** oder die Lage von Tools direkt in die Skripten ein. Der Runlevel-Mechanismus orientiert sich an der Linux Standard Base und an dem von SuSE, zumindest, was die Verwendung einer zentralen Konfigurationsdatei (*rc.config*) anbetrifft. Portierungen auf andere Distributionen betreffen in erster Linie Pfadanpassungen für aufgerufene Tools[29]. Im **hwsetup**-Skript muss entweder auf eine automatische Hardware-Erkennung verzichtet werden oder die entsprechenden Anpassungen auf Tools anderer Distributionen erfolgen. Die automatische Hardwareerkennung gehört inzwischen zum Standard moderner Linux-Distributionen, eventuell hilft auch ein Blick auf Knoppix[30] weiter.

Generell sollten auf dem Server folgende Dienste laufen bzw. installiert sein, damit sie automatisch in das Client-Root-Verzeichnis übernommen werden können. Je nach Bootimplementation kann das eine oder andere Programm entfallen oder zusätzlich notwendig werden:

- tftp - Trivial File Transfer Protocol Dienst, für Dateiübertragungen ohne Benutzeranmeldung. Dieses kommt auf dem Server zum Einsatz. Bei SuSE heißt das Paket *atftp.rpm*
- nfs-server - Es sollte besser der Userspace-NFS-Daemon verwendet werden, da bei hoher Belastung Probleme beim kernelbasierten NFS auftreten.

[29] siehe die Installationsskripten **mk_dxsfs.sh** und **mkdxsinittar**

[30] Ein CD-Only-Linux welches sich automatisch von der CD starten und benutzen läßt.

- devfsd - für das */dev* Dateisystem der Clients. Auf dem Server eventuell überhaupt nicht aktiv/benötigt. (*devfsd.rpm*)
- dhcp-client - Clientsoftware für DHCP-Anfragen, die nur auf den Clients benötigt wird. Andere Clientimplementationen als die ISC sind derzeit nicht geeignet. (*dhcp-client.rpm*)
- dhcp-server - Serversoftware für DHCP (Version 3, es gehören mehrere RPM-Pakete zum ISC-DHCP: *dhcp-base.rpm* und *dhcp-server.rpm*)
- etherboot und etherboot-mknbi (für die Bearbeitung der Client-Kernel)
- syslinux - für PXE-Boot (**gethostip** Bestandteil des *syslinux.rpm*, die zwingend notwendigen Dateien sind bereits im DXS-Dateibaum unterhalb von *boot* vorhanden.)
- busybox - erleichtert aufgrund der integrierten All-in-One-Lösung von Shell und Tools das Debugging im INITTAR. Funktioniert nicht mit allen Versionen! (Software *busybox.rpm*)

Der Start der Services NFS, TFTP, DNS und deren Funktion sollte anhand der jeweiligen Log-Dateien überprüft werden. Einzelne Dienste liefern eigene Test-Tools für solche Zwecke[31] mit.

C.2 Einrichtung des Client-Dateibaumes

Theoretisch sollte ein einfaches Auspaken der jeweils aktuellen Version der DXS-Umgebung, ein Wechseln in das entstandene Verzeichnis *linux-diskless* und Aufrufen von **./mk_dxsfs.sh** in diesem genügen. Die Fragen, welche das Skript stellt, können meistens einfach mit "Enter" beantwortet werden, die Beispielausgabe für die */etc/exports* kann als Vorlage zum Anlegen der Konfigurationsdatei des NFS-Servers verwendet werden. Am Ende des Skriptlaufes werden eventuell einige Fehlermeldungen angezeigt, die auf fehlende Programme oder Dienste hinweisen. In dieser Anleitung wird von */nfsroot/dxs* als Standard-Client-Root ausgegangen; dieses kann jedoch auch geändert werden. Generell gilt: Nur die Konfigurationsverzeichnisse, -dateien, Programme und Bibliotheken aus dem Basisdateisystem (ohne */usr*, */opt*) werden übernommen, die zum Zeitpunkt der Ausführung von **./mk_dxsfs.sh** installiert sind. Wird z.B. **gimp** nachträglich einer bestehenden Softwareauswahl hinzugefügt, müssen die Konfigurationsdateien händisch in das Client-Verzeichnis eingetragen werden oder das Installationsskript erneut laufen.

Nachdem die erste Installation erfolgt ist, sollte man die DHCP-Konfigurationsdatei anlegen. Ein Beispiel *dhcpd.example* befindet sich im Installationsverzeichnis und kann als Vorlage dienen. Bestimmte Optionen müssen gesetzt werden, damit der Client die Serverantwort akzeptiert. Dieses Zusammenspiel erfolgt durch die Einstellungen in der *dhclient.conf*, die sich unterhalb von */nfsroot/dxs/var/ram* befindet. Nachdem irgendwelche Änderungen unterhalb dieses Verzeichnisses erfolgt sind, muss zum Wirksamwerden **mkdxsinittar**[32] aufgerufen werden, welches die INITTAR mit den erfolgten Änderungen erneut generiert.

Überprüft oder angepasst werden müssen die Einstellungen für den Automounter, *auto.master* und *auto.home*, die verschiedenen Displaymanager, die Vorgaben für den X-Server und Kernel-Module in **hwsetup** und Dateien für spezielle Aufgaben, wie die PAM-Module[33].

[31] **dig** oder **nslookup** für DNS, NFS kann mittels **mount** am einfachsten auf dem Server selbst ausprobiert werden, ...

[32] zu Optionen konsultiere man den Skriptanhang

[33] Die Pluggable Authentication Modules erlauben differenzierte Authentifizierung und das Ausführen spezieller Aufgaben, wie eines Samba- oder Loopbackmounts für Cryptodateisysteme.

Die Einrichtung einzelner Dienste kann durchaus etwas aufwändiger ausfallen, da die besondere Situation des Client-Dateisystems mit */var* und */etc* insbesonderem beachtet werden muss. Nicht alle Verzeichnisse können wie gewohnt genutzt werden, was den korrekten Start verschiedener Dienste verhindern kann.

C.3 DHCP-Konfigurations-Optionen

Das Verhalten der Diskless Clients wird wesentlich durch DHCP-Optionen gesteuert, wobei vordefinierte und user-spezifische Optionen verwendet werden. Weitere Erklärungen finden sich in der Beispieldatei *dhcpd.example*.

DHCP-Option	Datentyp	Erklärung
root-path	String	Pfad des NFS-Root
dhcp-max-message-size	Integer	Maximale Größe eines DHCP-Datenpaketes
broadcast-address	IP-Address	Broadcast-Adresse für Clients
routers	List of IP-Addresses	Kommaseparierte Liste der Gateway-IP-Nummern
domain-name-servers	List of IP-Addresses	Kommaseparierte Liste der DNS-Server
domain-name	String	Space-separierte Liste der DomainAnhänge, der erste Eintrag zusammen mit Hostname FQDN des Clients
nis-servers	List of IP-Addresses	Kommaseparierte Liste der NIS-Server
nis-domain	String	NIS-Domain-String
font-servers	List of IP-Addresses	Kommaseparierte Liste der Fontserver für X11
x-display-manager	List of IP-Addresses	Kommaseparierte Liste der Server, welche mittels XDMCP X11-Logins anbieten
netbios-name-servers	List of IP-Addresses	Kommaseparierte Liste der WINS (eine Art DNS für Windowsnetzwerke)
ntp-servers	List of IP-Addresses	Kommaseparierte Liste der Network Time Protocol Server

Tabelle 1: Vordefinierte DHCP-Optionen

An der Stelle von IP-Listen können auch Hostnamen oder Full Qualified Domain Names (FQDN) stehen, wenn diese korrekt vom DHCP-Server in IP-Nummern aufgelöst werden können. Allen genannten Optionen muss der Identifikationsstring "option" vorangestellt werden, z.B. *option start-x "indirect";*. Alle hier verwendeten Bezeichner für die frei-wählbaren DHCP-Optionen können auch anders benannt werden. Die Zuordnung geschieht zu Beginn der jeweiligen Konfigurationsdateien, wie im Abschnitt zu DHCP ausgeführt wird.

Werden bestimmte Variablen nicht gesetzt, kommen in den meisten Fällen defaults zum Einsatz, die sich über die *dhclient.conf*, *rc.config* und die Setup-Skripten **dhclient-script** bzw. **hwsetup** bestimmen. Die Option "x-server-defs" kennt die space-separierten Einträge für Tastatur (de|us), für deutsche oder englische Tastatur, den Maustyp, den Mausport (psaux|ttySN|input/mice), die horizontale und vertikale Monitormaximalfrequenz, die maximale Bildschirmauflösung (640x480|800x600|1024x768|...), das XFree86-Modul für die Grafikkarte und die Farbtiefe in Bit (8|15|16|24).

Mittels "start-x" wird eingestellt, wie sich der X-Serververhält. Möglich sind eine direkte Verbindung auf den eigenen Displaymanager (direct|yes), eine direkte Verbindung auf den

DHCP-Option	Datentyp	Erklärung
o128	String	Magic-Field für Etherboot
o129	String	...
menudflts	String	Defaultauswahl für Etherboot-Menü
motdline1	String	Zeile für Message of the Day (motd) bei Etherboot-Menüs
menuline1	String	Erste Menüzeile für Etherboot
menuline2	String	Zweite Menüzeile für Etherboot
menuline3	String	Dritte Menüzeile für Etherboot
bootlocal-script	String	Name eines Skriptes oder eines Befehls zur Ausführung wärend des Bootens
language	String	Einstellung der Standardsprache
start-x	String	Bestimmt Verhalten des X-Servers
start-snmp	String	Bestimmt ob SNMP gestartet wird (yes|no)
start-sshd	String	Bestimmt ob SSH gestartet wird (yes|no)
start-xdmcp	String	Bestimmt ob und welcher Displaymanager gestartet wird
start-cron	String	Bestimmt ob CRON gestartet wird (yes|no)
crontab-entries	String	Erlaubt *crontab*-Einträge zu generieren
start-rwhod	String	Bestimmt ob der RWHO-Dienst gestartet wird (yes|no)
start-printdaemon	String	Bestimmt ob und welcher Druckservice aktiviert wird
tex-enable	String	Bestimmt darüber ob bestimmte Zusatzverzeichnisse für LaTex gemountet werden
netbios-workgroup	String	Setzt den Arbeitsgruppennamen für Windowsnetzwerke
start-vmkernel	String	Bestimmt über das Laden der Kernelmodule für den Betrieb von VMware (yes|no)
hw-mouse	String	Hardwaredefinition der Maus für XF86Config und gpm
hw-graphic	String	Hardwaredefinition der Grafikkarte (Modul)
hw-monitor	String	Hardwaredefinition des Monitors (Maximalfrequenzen und Auflösung)

Tabelle 2: Benutzerdefinierte DHCP-Optionen

Chooser des eigenen Displaymanagers (indirect), das Starten von KDE, Gnome, ... ohne den Umweg über ein grafisches Login (kde|gnome|fvwm|...) und das Starten von VMware oder Citrix-Metaframe ohne Windowmanager (vmware|citrix) und grafische Login-Aufforderung.

Die Optionen "o128", "o129", "menudflts", "motdline1", "menuline1", "menuline2" und "menuline3" sind etherboot-spezifisch und können durch geeignete DHCP-Server-Konfiguration doppelt belegt werden. Wenn sich Etherboot meldet, wird eine Antwort, wenn sich **dhclient** meldet, eine entsprechende andere Serverantwort abgesetzt. Dieses Verfahren wird bereits für den Ketten-Start von PXE und Etherboot eingesetzt. "start-printdaemon" entscheidet darüber welcher Druckdienst (cups|lpd|no) gestartet wird, wobei derzeitig CUPS oder LPD mögliche Optionen sind. Die entsprechende Konfigurationsdatei für CUPS im Client-Modus wird mittels **dhclient-script** generiert.

C.4 Vorbereitung der Client-Rechner

Sicherlich werden sich einige eingangs genannte Schritte an dieser Stelle in ihrer Nennung wiederholen; hier geht es jedoch nur um den Schnelleinstieg. Der typische LDC-Rechner ver-

fügt über eine bootfähige Netzwerkkarte, entweder mit einem Etherboot-ROM oder einer bereits vorhandenen PreBoot Extension(PXE). Letzte muss evtl. noch im BIOS des Clients eingeschaltet und in der Bootreihenfolge geeignet eingetragen werden. Das Etherboot muss nicht zwingend in einem separaten EPROM-Baustein untergebracht, sondern kann dem Mainboard-BIOS unter Umständen hinzugefügt werden. Eine ausführliche Beschreibung zu den einsetzbaren EPROMs und dem Verfahren des BIOS-Patchens entnehme man den Ausführungen in [9].

C.5 Hardware-Unterstützung und Konfiguration

In jedem Fall sollte sichergestellt sein, dass alle benötigten Hardware-Komponenten des Clients unterstützt sind. Dieses beginnt mit Etherboot, welches für den entsprechenden Ethernet-Chip vorliegen muss. Bei PXE entfällt dieses Problem. Innerhalb des INITTAR werden vom **init** nacheinander die verschiedenen Kernelmodule für die Ethernet-Chips durchprobiert. Liegen mehrere Implementationen für einen Typ von Chip vor, wie z.B. bei den RTL8139A,B,C,D-basierten Netzwerkadaptern, kann entweder *8139too.o* oder *rtl8139.o* verwendet werden. Dieses muss im **init** geeignet eingestellt werden. Am besten sollte die Reihenfolge der durchprobierten Module nach der Häufigkeit des jeweiligen Netzwerkadapters gewählt werden:

```
[...]
for module in 3c59x \
        8139too         tulip           natsemi         sis900  \
        via-rhine       ne2k-pci        dmfe            epic100 \
        winbond-840     eepro100        e2100           cs89x0  \
        bcm4400         tg3             e1000                   \
        lance           pcnet32         smc9194         3c509   \
        wd              smc-ultra       eexpress        ni65    \
        "ne io=0x300"   "ne io=0x320"   "ne io=0x340"           \
        "ne io=0x360"   "ne io=0x200"   "ne io=0x220"           \
        "ne io=0x240"   "ne io=0x250"   "ne io=0x260"           \
        "ne io=0x280"   \
        depca           eepro           ni5010          ni52    \
        3c515           3c507; do
        [ $DEBUG -gt 0 ] && echo "Trying module: $module"
        modprobe -q $module >/dev/null 2>&1 && break
done
[...]
```

Weiterhin muss die Grafikkarte unterstützt werden. Dieses betrifft sowohl die Auswahl des geeigneten XFree86-Moduls als auch die Monitorauflösungen, Farbtiefen und Wiederholfrequenzen. Festfrequenzmonitore und TFTP-Displays haben zum Teil recht enge Spezifikationen. Die entsprechenden ModLines sind zu Beginn der **hwsetup** bei Bedarf anzupassen. Nähere Ausführungen zur optimalen Konfiguration von XFree86 kann man in [9] nachlesen. Je nach Anwendungszweck sind externe Schnittstellen, wie USB oder IEEE1394 von geringerer Bedeutung. Die notwendigen Kernel-Module werden gleichfalls in der **hwsetup** geladen, bei Problemen sollte hier die Reihenfolge oder die Art des Moduls[34] geändert werden.

[34] Auswahl ob *usb-uhci.o* oder *uhci.o* eingesetzt wird

C.6 X11-Konfiguration

Die Diskless Clients können in einer ganzen Reihe von Betriebsarten eingesetzt werden, welche z.T. mittels **start-gui** und/oder *inittab* gesteuert werden. Wird der X-Server direkt gestartet, um die Maschine direkt ins KDE, eine VMware-Session oder den Citrix-Metaframe-Client zu booten, muss kein Displaymanager[35] laufen. Wird jedoch der Displaymanager bemüht[36] muss dieser entsprechend konfiguriert, d.h. das XDMCP[37] eingeschaltet sein.

Hierfür sind jeweils die Konfigurationen der Displaymanager anzupassen. Deren Konfigurationsdateien sollten möglichst unterhalb des */etc*-Verzeichnisses abgelegt sein und nicht als Hardlink auf die entsprechende Datei des Servers existieren. Bei der SuSE-Distribution findet man die Einstellungen für den **kdm** unterhalb von */etc(.s)/opt/kde(2|3)/share/config/kdm*. Eine wichtige Auswahl von Optionen der *kdmrc* findet sich im nachstehenden Beispiel:

```
[...]
Enable=true
Port=177
ChoiceTimeout=10
Xaccess=/etc/X11/xdm/Xaccess
Willing=/etc/X11/xdm/Xwilling
[...]
```

Der **gdm** läuft aus Sicherheitsgründen nicht unter der *root*-ID, der gewählte alternative Nutzer und die Gruppe müssen in den entsprechenden Unix-Dateien *passwd* und *groups* vorhanden sein. Die Rechte für eingestellte temporäre Verzeichnisse des **gdm** müssen stimmen, da der Dienst sonst unter Fehlermeldung abbricht. Die Konfiguration erfolgt in der *gdm.conf*, die sich z.B. unterhalb von */etc(.s)/opt/gnome/gdm* findet:

```
[...]
User=gdm
Group=shadow
LogDir=/var/lib/gdm
[...]
[xdmcp]
Enable=true
HonorIndirect=true
MaxPending=4
MaxPendingIndirect=4
MaxSessions=16
MaxWait=15
MaxWaitIndirect=15
DisplaysPerHost=1
Port=177
Willing=/etc/X11/xdm/Xwilling
[...]
[servers]
# comment all entries out!
```

[35] mögliche Displaymanager sind **kdm**, **xdm**, **wdm**, **gdm**, welche sich in ihrer Konfiguration mehr oder weniger unterscheiden können

[36] Anders als auf klassischen Desktopmaschinen übernimmt er jedoch nicht die Aufgabe dafür zu sorgen, dass immer ein X-Server läuft

[37] das X-Display Manager Control Protocol arbeitet auf UDP Port 177 und tauscht darüber Managementinformationen für X-Displays aus

Für besonders schlanke Installtionen und ältere Geräte möchte man vielleicht den **xdm** einsetzen. Dessen Konfigurationsverzeichnis ist üblicherweise */etc(.s)/X11/xdm*, die Konfigurationsdatei *xdm-config*. Für den **wdm** benötigt man eine nur um wenige Optionen erweiterte Konfigurationsdatei, die meisten Einträge sind jedoch identisch:

```
[...]
DisplayManager.accessFile:      /etc/X11/xdm/Xaccess
DisplayManager.*.session:       /etc/X11/xdm/Xsession
DisplayManager.requestPort:    177
[...]
! extensions for wdm display manager
DisplayManager*wdmWm:           icewm:wmaker:gnome:kde:fvwm
DisplayManager*wdmRoot:         false
DisplayManager*wdmAnimations:   true
[...]
```

Einige Standardkonfigurationsdateien übernehmen die aktuellen Windowmanager **gdm** und **kdm** direkt, hierzu zählt z.B. die *Xaccess*. Diese Datei wird durch **dhclient-script** beim Booten des Clients eingerichtet. Andere Dateien, wie *Xservers* oder *Xsession* müssen explizit angegeben werden, weil sonst die Default-Pfade abweichen können. Für die DXS-Umgebung sollte die *Xservers* keine aktiven Zeilen enthalten, da der X-Server per System-Init, analog zu den Text-Consolen gesteuert wird.

C.7 Anwendungsprogramme

Jede in der DXS-Umgebung genutzte Software muss auf ihre "Verträglichkeit" hin untersucht werden:

- Wohin werden Komponenten der Software installiert: Ausschließlich */opt* oder */usr* ist unproblematisch und stellt die Software ohne Neukonfiguration der DXS-Umgebung bereit. Anpassungen für */etc* oder */var* erfordern evtl. mehr Arbeit und eine Neuinitialisierung durch den Aufruf von **mk_dxsfs.sh**.
- Gibt es spezielle Ausführrechte, die den normalen Benutzer einschränken: Dann sind Programme im Bereich */opt* oder */usr* unter Umständen nicht für den Benutzer "root" ausführbar, oder die Option "no_root_squash" muss in die Konfigurationsdatei des NFS-Servers, *exports* eingefügt werden.
- Wird viel Platz im */tmp* oder Bereichen des */var* belegt: Wenn viele Daten in die Ramdisk geschrieben werden, kann diese überlaufen und damit den gesamten Client stilllegen. Hier kann Abhilfe geschaffen werden, wenn in einen schreibbaren Bereich des NFS, z.B. */tmp/scratch* verlinkt wird.
- Gibt es lizenzrechtliche Einschränkungen bei der Verbreitung: Einmal installierte Software kann meistens sehr bequem auf allen angeschlossenen Clients gestartet werden. Entweder man wählt die Software von vorneherein nach dem besten Lizenzmodell aus, beschafft geeignete Lizenzen, einen Lizenzmanager oder regelt das Problem über eingeschränkte NFS-Freigaben.

Für Programme gibt es drei wesentliche Unterschiede für ihre Konfigurationsdateien: Sie verwenden keine systemweiten Einstellungen in */etc* und sind damit völlig unproblematisch. Sie benutzen eine systemweite Einstellung, welche zwischen dem Server und allen Clients

identisch ist. Dann muss lediglich die Konfiguration kopiert oder verlinkt werden. Server und Clients verlangen verschiedene Einstellungen, diese sind aber für alle Clients gleich: Dieses gilt z.B. für die Displaymanager. Die Konfigurationsdateien müssen ins Client-Root kopiert und angepasst werden. Am aufwändigsten sind Szenarien, wenn eine Anpassung per Client erfolgen soll. Diese kann z.B. durch **dhclient-script**. Eventuell sind Templates geeignet dem INITTAR unterhalb von */nfsroot/dxs/var/ram/etc* zur Verfügung zu stellen.

D Weiterführende Information

D.1 Print

[1] Ralph Droms,Ted Lemon, *The DHCP Handbook: Understanding, Deploying, and Managing Automated Configuration Services*, New Riders Publishing, 1999.

[2] Bauer, Günter, "Über 100 Uni-Rechner im Griff", *NetworkWorld* 26. Oktober 2001, Seite 28.

[3] Hantelmann et. al., "Jenseits des PC's", *iX* 03/2000, Seiten 56-68,130-136.

[4] Ritter, Marcel, "Automatische Installation und Konfiguration von Linux", *Linux-Magazin* 09/2001, Seite 97-101.

[5] von Suchodoletz, D., "Gut gebootet, Linux X-Terminals", *Linux-Magazin* 08/1999, Seite 101-110.

[6] von Suchodoletz, D., "Thin-Clients - Plattenloser Arbeitsplatz selbstgemacht", *Linux-Magazin* 08/2000, Seite 110-115.

[7] Kulisch, Meyer, Steffens, "Ausgetauscht, Linux auf 3000 verteilten Arbeitsplätzen", *iX* 03/2002, Seite 40 ff.

[8] von Suchodoletz, D., "Die Netzstarter, Diskless Clients mit Linux - eine Handlungsanleitung", *Linux-Magazin* 01/2003, Seite 74-81.

[9] von Suchodoletz, D., *Effizienter Betrieb großer Rechnerpools, Implementierung am Beispiel des Studierendennetzes an der Universität*, GwDG Göttingen, 2003.

D.2 Web

[W1] http://www.thinguin.org

[W2] http://www.rom-o-matic.net

[W3] http://etherboot.sourceforge.net

[W4] http://netboot.sourceforge.net

[W5] http://www.ltsp.org

[W6] http://www.escape.de/users/outback/linux/dlc.html

[W7] http://ldc.goe.net