

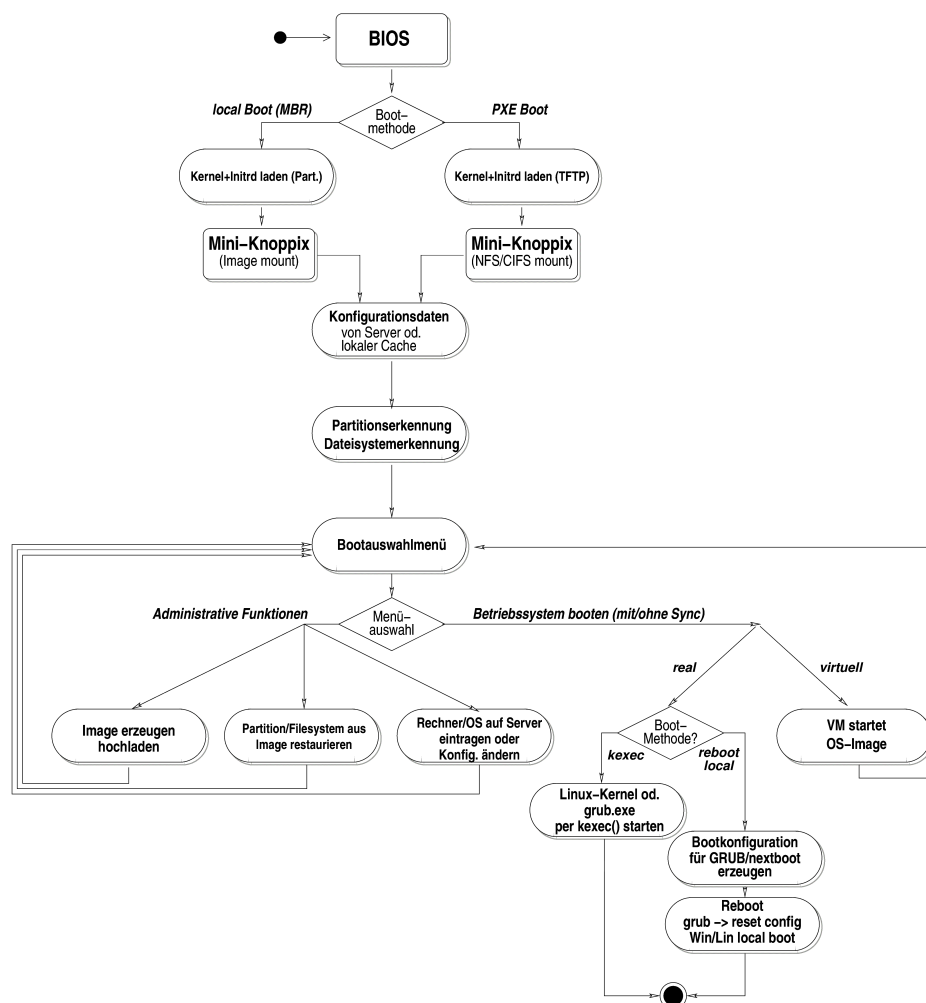
LINBO V3

(**L**inux-basiertes **I**nteraktives **N**etzwerk-**B**ootsystem)

Technische Dokumentation und Handbuch für IT-Personal

Dipl.-Ing. Klaus Knopper (Grundlagen, Technik, Backend+API)
Dipl.-Inf. Martin Öhler (GUI Client+Server)

Stand: 1. Juni 2014



Inhaltsverzeichnis

1	Einleitung.....	4
1.1	Funktionsweise / Technik.....	4
1.2	Testlauf innerhalb des Buildsystems per qemu/kvm (für Entwickler).....	5
1.3	Namen und Beschreibungen.....	5
1.3.1	Cache-Partition.....	5
1.3.2	Multicast.....	6
1.3.3	Torrent.....	6
1.3.4	PXE.....	6
1.3.5	cloop.....	6
1.3.6	rsync.....	7
2	Installation / Inbetriebnahme von LINBO.....	8
2.1	LINBO-Server.....	8
2.1.1	PXE-Konfiguration (DHCP-Server).....	8
2.1.2	RSYNC-Konfiguration (Server).....	10
2.1.3	NFS und/oder SAMBA-Server.....	12
2.1.4	LINBO-eigene Daten und Dateien.....	14
2.2	LINBO-Clients.....	15
2.2.1	Bootvorgang.....	15
2.2.2	LINBO- und Dateisystem-Konfiguration.....	16
3	Anwendung von LINBO.....	17
3.1	LINBO booten.....	17
3.2	Graphische Oberfläche von LINBO.....	18
3.3	Betriebssysteme wiederherstellen und starten.....	20
3.3.1	Button „Start“.....	20
3.3.2	Button „Sync+Start“.....	20
3.4	Betriebssystem-Images erzeugen und verwalten / Admin/IT-Personal Rolle.....	21
3.5	Der LINBO „Master-Modus“.....	23
3.6	Serverseitige Konfiguration / Client-Aufnahme.....	25
3.6.1	Expert Configuration.....	25
3.6.2	Client Wizard.....	35
3.6.3	Image Verwaltung per GUI.....	38
4	Konfiguration von LINBO und Betriebssystemen.....	42
4.1	start.conf - Partitionen und Images.....	42
4.1.1	Abschnitte in start.conf- Übersicht.....	42
4.1.2	Abschnitt [LINBO].....	43
4.1.3	Abschnitt [Disk].....	44
4.1.4	Abschnitt [Partition].....	44
4.1.5	Abschnitt [OS].....	45
4.1.6	Abschnitt [VM].....	47
4.1.7	Beispiel für eine gültige start.conf-Datei.....	48
4.1.8	Windows-Patches - .reg-Dateien.....	50
4.1.9	Upload-Dateirechte automatisch korrigieren.....	51
5	Das LINBO-Kochbuch, „How do I ...?“.....	52
5.1	Der allererste Start: Wie richte ich ein Master-System für die Erzeugung	

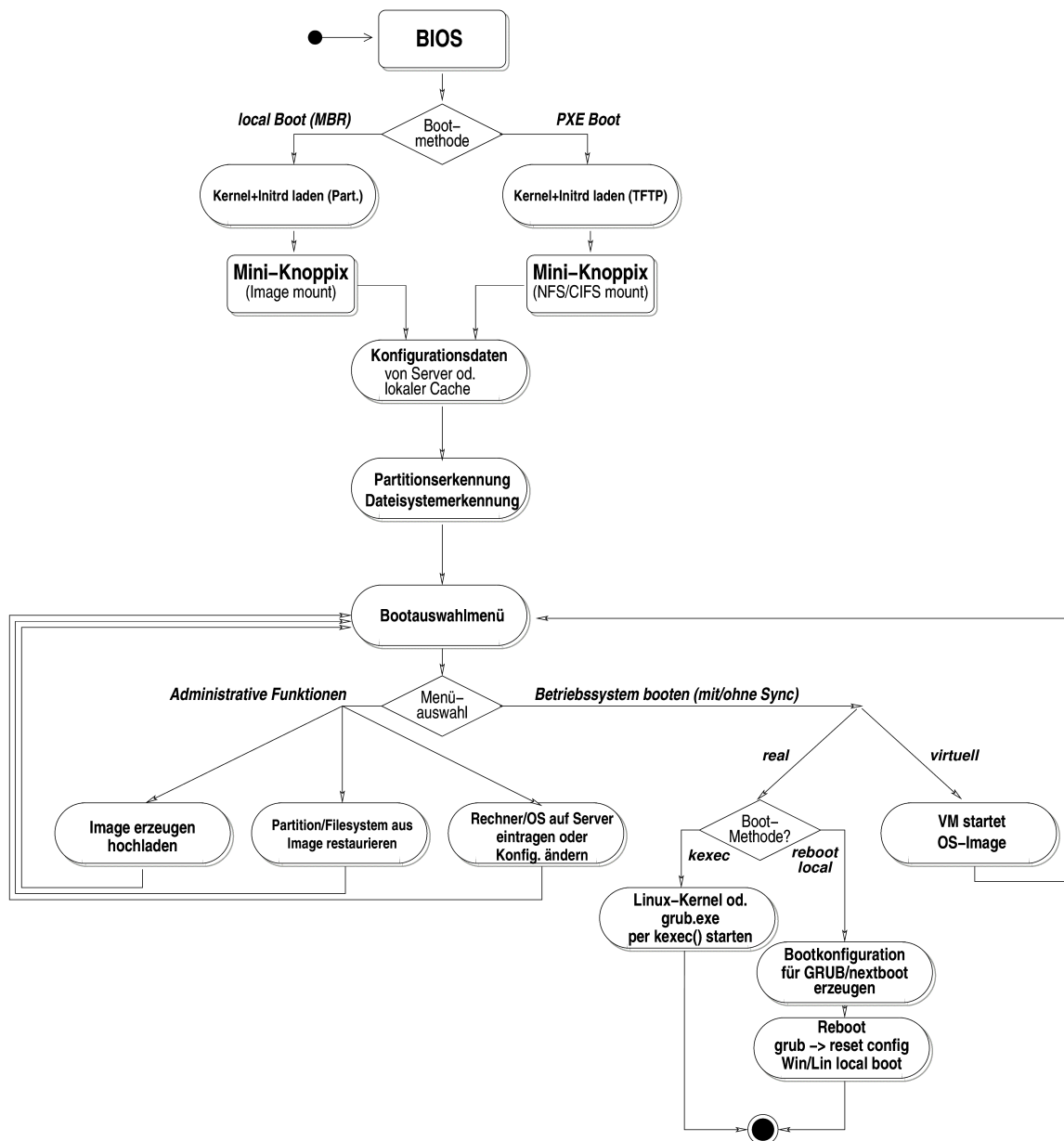
von LINBO-Images ein?.....	52
5.2 Wie groß soll die Cache-Partition sein, und wo genau soll sie auf der Festplatte liegen?.....	53
5.3 Wie setzen sich die Partitionsnamen unter Linux zusammen, was muss ich angeben?.....	53
5.4 Welche Dateisystem-Typen muss ich in start.conf angeben? (FSType = ?).....	54
5.5 Hilfe, es bootet nicht!.....	55
5.5.1 LINBO startet nicht / bleibt stehen.....	55
5.5.2 Das von LINBO gestartete Betriebssystem startet nicht / bleibt stehen.....	58
5.6 Wie erzeuge ich ein virtuelles System mit und für virtualbox, zur Benutzung durch LINBO?.....	60
6 LINBO Backend – API.....	62
7 rsync - physikalische Limits.....	65

1 Einleitung

LINBO ist ein halb- bis vollautomatisch (je nach Konfiguration) arbeitender Bootmanager, der verschiedene auf Festplatte installierte Betriebssysteme oder virtuelle Maschinen starten kann, und Wartungs- und Reparaturfunktionen für die schnelle Installation und Instandsetzung von Computersystemen zur Verfügung stellt.

Basierend auf Knoppix™-Technologie ist Version 3 ein vollwertiges Betriebssystem mit Anwendungen, das im Live-Betrieb von einer „Cache-Partition“ oder per PXE gebootet wird.

1.1 Funktionsweise / Technik



Im Netzwerkbetrieb wird üblicherweise der Kernel und eine initiale Ramdisk (initrd) vom PXE/DHCP/TFTP-Server ins RAM auf den Client geladen, das komprimierten LINBO-Dateisystem (ca. 700MB) wird per SMB oder NFS eingebunden und im lokalen Cache, sofern vorhanden, wird eine aktualisierte Kopie des Systems angelegt.

Im lokalen („offline“) Betrieb werden alle Dateien hingegen direkt von einer „Cache-Partition“ per grub-Bootloader verwendet.

Nach normalerweise recht kurzer Startzeit wird eine graphische Oberfläche basierend auf HTML5/Ajax/Django mit Unterstützung eines lokal laufenden Apache-Webserver sowie Firefox präsentiert. Nach Auswahl eines Buttons werden verschiedene Aktionen über das Worker-Backend `linbo_cmd` abgewickelt.

Der Bauvorgang des LINBO-Systems sowie verschiedene Test-Szenarien werden in der LINBO-Entwicklungsumgebung mit Hilfe eines für Debian GNU/Linux ausgelegten Makefile durchgeführt.

1.2 Testlauf innerhalb des Buildsystems per qemu/kvm (für Entwickler)

(Installation in einer realen DHCP-Server-Umgebung: Siehe Abschnitt 8 auf Seite 8)

Das zentrale Makefile im LINBO-Entwicklungsverzeichnis bietet zwei Testszenarien mit Hilfe von kvm als Virtualisierer:

<code>make pxetest</code>	Start des internen DHCP, TFTP- und SMB-Servers, Booten von LINBO per PXE
<code>make hdttest</code>	Start von LINBO von virtueller Festplatte

1.3 Namen und Beschreibungen

1.3.1 Cache-Partition

Auf den mit LINBO verwalteten Rechnern wird eine LINBO-eigene Cache-Partition verwendet, i.d.R. ist dies die letzte Partition, die sich über den noch verbleibenden freien „Rest“ der Festplattenkapazität erstreckt). Hier werden für den lokalen Start (mit oder ohne Netzwerk) die von LINBO verwalteten Betriebssysteme und das LINBO-System selbst vorgehalten. Beim ersten Start

des Rechners per PXE werden bei entsprechend gesetzten Konfigurationsoptionen sowohl die Betriebssysteme als auch LINBO per RSYNC, Multicast oder Torrent auf die Cache-Partition heruntergeladen.

1.3.2 Multicast

Um den Cache vieler PCs mit den großen Image-Dateien (komprimiert ca. 500MB-80GB pro Betriebssystem je nach Ausstattung) effizient zu füllen, kann optional Multicast verwendet werden. Hierzu muss auf dem LINBO-Server udpcast, v.a. der udp-server installiert sein, welcher nach einer einstellbaren Mindestanzahl anfordernder Clients und ebenfalls einstellbarer Wartezeit das Senden der Images an mehrere Rechner gleichzeitig unterstützt. Dadurch werden die Daten nur einmal physikalisch übertragen, auch wenn mehrere Clients gleichzeitig den Cache mit Daten füllen, wodurch der Zeitaufwand beim erstmaligen Installieren minimiert wird.

Bei individuellen Clients wird hingegen stets auf dem Server nach einem Update des gewählten Images geprüft, und die gegenüber der älteren Version werden per RSYNC übertragen. Sind keine Aktualisierungen vorhanden, so wird die Version aus dem Cache weiterverwendet.

1.3.3 Torrent

Eine Alternative zur „Server-zentrischen“ Verteilung von Images bietet bittorrent. Hierbei stellen sich die Client-Rechner gegenseitig Images zur Verfügung, der Server dient lediglich als „Tracker“ um die Clients untereinander zu koordinieren. Dieses experimentelle Feature lohnt sich in einem „geswitchten“ Netzwerk, um den zentralen Server zu entlasten.

1.3.4 PXE

Pre Execution Environment“ bezeichnet eine standardisierte Methode, ein Bootmenü oder Betriebssystem übers Netzwerk zu laden und zu starten. Hierfür ist entweder eine PXE-fähige Netzwerkkarte mit entsprechender Firmware erforderlich (die in den meisten modernen Rechnern verfügbar ist), oder eine entsprechende Bootdiskette mit Treibern von <http://www.rom-o-matic.net>.

1.3.5 cloop

Das „*Compressed Loopback Device*“ ist ein von iptables-Autor Paul Russel und Klaus Knopper entwickeltes Block-Device Kernelmodul, das typischerweise eine Festplattenpartition in komprimierter Form in einer Datei verwendet. In

LINBO besitzen diese Dateien die Endung `.cloop`. Im Gegensatz zu den bekannten `zip` oder `tar.gz`-Archiven verhält sich ein über das `cloop`-Device eingebundenes Archiv wie eine Festplattenpartition mit wahlfreiem Zugriff, die enthaltenen Daten bzw. Teile davon werden aber erst beim Zugriff im Speicher dekomprimiert. In diesem Dateiformat ist es möglich, sämtliche Zusatzdaten wie Boot-Records oder „versteckte“ Informationen als 1:1 Kopie Betriebssystem- und Dateisystem-unabhängig zugänglich zu halten. Insbesondere das Herauskopieren einzelner Dateien ist möglich, ohne das gesamte Archiv auspacken zu müssen. In LINBO werden alle Images (direkte Partitionsabzüge) in diesem Format komprimiert gespeichert, was auf der Cache-Partition Platz spart und den Lesevorgang dadurch, dass weniger physikalische Lesezugriffe erfolgen müssen, stark beschleunigt. Dieses Verfahren ist auch von der KNOPPIX-DVD bekannt. Die Kompressionsrate beträgt bei ausführbaren Programmen zirka 3:1, bei Textdateien bis 12:1, bei Zufallsdaten oder verschlüsselten Dateien oder bereits komprimierten Bildern kann der Platzbedarf sich sogar geringfügig vergrößern.

1.3.6 rsync

`rsync` ist ein Synchronisierungs-Programm, das zwar wie viele Kopierprogramme auch zunächst eine 1:1 Kopie erzeugt, wobei aber tatsächlich nur die Änderungen zwischen Quelle und Ziel übertragen werden.

Die für LINBO verwendete Version von `rsync` enthält einen Patch, der bestimmte für das unter Windows verwendete NTFS-Dateisystem notwendigen Systemattribute erkennen mitkopieren kann. Im Entwicklungssystem sind die entsprechenden Quelltexte unter `Sources/rsync-*` untergebracht.

2 Installation / Inbetriebnahme von LINBO

Da LINBO auf der Client-Seite für die Erstinstallation lediglich einen per PXE/Netzwerkkarte bootfähigen Rechner mit Intel oder AMD-kompatibler Architektur voraussetzt, konzentrieren sich die Arbeiten für die Erstinstallation auf den Server, der die Clients mit IP-Adressen und Netzwerkspeicherkapazität versorgt.

2.1 LINBO-Server

LINBO setzt an Infrastruktur-Diensten voraus:

1. Einen DHCP-Server mit aktivierter PXE-Boot Unterstützung,
2. Einen TFTP-Server, der den Bootlader (`pxelinux`), dessen Konfigurationsdateien sowie den in LINBO enthaltenen Linux-Kernel (`linux` bzw. `linux64`) und eine initiale Ramdisk (`minirt.gz`) zum Download für die Clients zur Verfügung stellt,
3. Einen RSYNC-Server, der den Abruf von Images sowie das passwortgeschützte Hochladen und Überschreiben von Images auf einer ausreichend großen Partition unterstützt,
4. Wahlweise einen NFS- oder SMB-Server, über den das komprimierte LINBO-Dateisystem für die initiale Installation direkt auf den Clients eingebunden werden kann,
5. (Optional) einen Apache-Webserver mit installiertem Django-Framework, um das serverseitige GUI für die Administration und Wartung der Images, Rechnergruppierung und Anlegen der Konfigurationsdateien für IT-Personal komfortabler durchführen zu können (manuelles Anlegen und Änderungen von Konfigurationsdateien ist alternativ für fachkundiges IT-Personal auch möglich),
6. (Optional) einen Multicast-Server zum synchronen Übertragen von Images an mehrere Clients gleichzeitig (`udpcast`),
7. (Optional) einen Torrent-Tracker, über den sich Clients gegenseitig Images unter optimaler Ausnutzung der verfügbaren Bandbreite untereinander abgleichen können.

2.1.1 PXE-Konfiguration (DHCP-Server)

LINBO-Kernel (`linux`, `linux64`) und initiale Ramdisk (`minirt.gz`) müssen sich in einem per TFTP erreichbaren Verzeichnis auf dem Server befinden. Ein typischer Name für dieses Verzeichnis ist auf vielen Unix-Systemen `/tftpboot`.

Der für LINBO empfohlene TFTP-Server `tftpd-hpa` kann beispielsweise auf dem Server wie folgt gestartet werden, wenn Kernel und Initial Ramdisk im

Verzeichnis, anders als zuvor angegeben, /var/linbo liegen:

```
sudo tftpd -l /var/linbo
```

Im DHCP-Server sind dann die Clients bzw. Client-Netze anzugeben, die per LINBO verwaltet werden sollen. Optional können für verschiedene Rechner auch entsprechend verschiedene Kernel oder initiale Ramdisks in der Konfiguration von pxelinux.cfg/CLIENT-ADRESSE (in Hexadezimalschreibweise) angegeben werden, in denen sich jeweils eine andere start.conf-Konfigurationsdatei befinden kann.

Beispiel für einen entsprechenden Abschnitt aus der dhcpd.conf des ISC-dhcpd Version 3:

```
allow booting;
allow bootp;

subnet 10.0.2.0 netmask 255.255.255.0 {
    next-server 10.0.2.2;
    filename "pxelinux.0";
    option subnet-mask 255.255.255.0;
    range 10.0.2.10 10.0.2.15;
    option domain-name-servers 10.0.2.2;
    option routers 10.0.2.2;
}
```

In diesem Beispiel werden die IP-Adressen 10.0.2.10 bis einschließlich 10.0.2.15 dynamisch vergeben, die Clients starten per TFTP den PXE-Bootlader pxelinux.0, der seine Konfigurationsdatei unter pxelinux.cfg/default nachlädt, sofern keine IP-spezifische Konfigurationsdatei existiert. LINBO-Kernel sowie die initiale Ramdisk müssen ebenfalls per TFTP erreichbar sein.

Hinweis: Üblicherweise fügen die Clients ein Pfad-Präfix / zum Dateinamen hinzu, daher sollte mit tftp server-ip getestet werden, ob der LINBO-Kernel per

```
get /linux
```

vom Server downloadbar ist.

2.1.2 RSYNC-Konfiguration (Server)

Zur Synchronisation von Images sowie zum Upload neu erzeugter Images wird rsync (vergl. Abschnit 1.3.6) verwendet.

Unter Debian wird rsync installiert als root mit dem Kommando `apt-get install rsync`. Damit die Clients Zugriff auf die Images bekommen, muss zunächst eine rsync-Freigabe [linbo] in `/etc/rsyncd.conf` auf dem Server eingerichtet werden:

```
[linbo]
comment = LINBO Image directory (read-only)
path = /srv/linbo
use chroot = no
lock file = /var/lock/rsyncd
read only = yes
list = yes
uid = nobody
gid = nogroup
dont compress = *.cloop *.rsync *.gz
```

Dieses Beispiel erlaubt einen **nur lesenden** Zugriff auf die Dateien im Verzeichnis `/var/linbo` für alle Clients ohne Passwort. Mit

```
rsync server-adresse::linbo
```

kann sich der fortgeschrittene Anwender das Verzeichnis testweise per rsync auflisten lassen, ohne eine Datei tatsächlich übertragen zu müssen.

Für die Übertragung von Images vom Client-Rechner zum Server, z.B. für neu erstellte Images, ist außerdem die Einrichtung eines schreibbaren rsync-Repository erforderlich. Der entsprechende zusätzliche Eintrag in `/etc/rsyncd.conf` lautet:

```
[linbo-upload]
comment = LINBO Upload directory
path = /var/linbo
use chroot = no
lock file = /var/lock/rsyncd
read only = no
list = yes
uid = root
gid = root
dont compress = *.cloop *.rsync *.gz
auth users = linbo
secrets file = /etc/rsyncd.secrets
```

Das tatsächliche Verzeichnis im Dateisystem ist in diesem Beispiel wieder das Verzeichnis `/var/linbo`. Dort sollte sich ebenfalls der Linbo-Kernel und die initiale Ramdisk für Updates des lokalen Client-Bootsystems befinden, sowie die Image-Dateien mit den Betriebssystemen, Partitionsdumps und virtuellen Maschinen (Abschnitt 4.1.6), für die Clients. Mit (Beispiel)

```
rsync datei.txt linbo@server-adresse::linbo-upload
```

kann eine Testdatei (`datei.txt`) an den Server übertragen werden (allerdings funktioniert dies erst nach dem nächsten Konfigurationsschritt). Hierbei sollte nach einem Passwort gefragt werden, was auch der Authentifizierung des „Administrators“ in LINBO dient.

Dieses Login/Passwort-Paar für die rsync-Freigabe `linbo-update` muss noch eingetragen werden, in die in der Konfigurationsdatei bereits angegebene Datei `/etc/rsyncd.secrets`.

Beispiel:

```
linbo:geheim
```

Das Passwort ist für die Sicherheit des LINBO-Systems essentiell, und sollte nur den Administratoren, die LINBO-Clients erstmalig aufsetzen und auch neue Images auf dem Server einspielen dürfen, bekannt sein. Für den normalen Betrieb von LINBO, also das Aktualisieren und Booten von Betriebssystemen auf LINBO-Clients, ist das Passwort nicht erforderlich.

Bitte beachten Sie, dass die Datei `/etc/rsyncd.secrets` nur für den rsync-Server lesbar sein darf, sonst verweigert rsync jedes Passwort. Mit dem

Linux-Kommando (als Administrator)

```
chmod 400 /etc/rsyncd.secrets
```

sollte dies gewährleistet sein.

Fehlermeldungen, Warnungen und Statusinformationen von rsync finden Sie auf den meisten Linux-Distributionen in den Logdateien `/var/log/syslog` oder `/var/log/daemon.log`.

Falls der rsync-Server die Änderungen an seiner Konfiguration nicht automatisch erkennt, muss er neu gestartet werden:

```
/etc/init.d/rsync restart
```

Ist der rsync-Server konfiguriert, so müssen noch der LINBO-Kernel, die initiale Ramdisk, eine Kopie des LINBO-Hauptdateisystems (Ordner LINBO mit Inhalt) sowie für das lokale, native Booten von Betriebssystemen `grub.exe` in das LINBO-Verzeichnis (in unserem Beispiel `/var/linbo`) kopiert werden.

Damit LINBO diese Daten bei einer Aktualisierung des Clients nicht jedesmal erneut herunterlädt, sollten auch die zu den genannten Dateien passenden `.info`-Dateien kopiert bzw. erzeugt werden, wie in diesem Beispiel für den 64bit Kernel `linux64`:

```
[linux64]
timestamp=201307251505
imagesize=4198
```

In diesen ist ein Zeitstempel und die Dateigrößen der Dateien vermerkt, so dass der zeitaufwändige Download der großen Dateien ggf. von LINBO übersprungen werden kann, wenn die Dateien im Cache noch aktuell sind.

Das gleiche Verfahren wird auch bei den großen Image- und VM-Dateien angewandt. Hier erzeugt LINBO clientseitig automatisch die entsprechenden `.info`-Dateien und lädt sie mit hoch auf den rsync-Server.

2.1.3 NFS und/oder SAMBA-Server

Da in LINBO V3 die zum Einsatz kommenden Werkzeuge inklusive graphischer Oberfläche sehr umfangreich sind (fast 2GB unkomprimiert), wird die Systemsoftware von LINBO nicht per TFTP oder RSYNC übertragen, sondern in komprimierter Form über ein Remote Dateisystem direkt eingebunden, welches von einem per Option angegebenen Server stammt.

Hierdurch werden nur die aktuell benötigten Daten und Programme, bandbreitensparend in cloop-komprimierter Form, übertragen. Sobald LINBO nach einer initialen Partitionierung und Einrichten einer Cache-Partition auf den Client kopiert wurde, ist das Einbinden auch von dieser Cache-Partition möglich. Im „Offline“-Modus, also ohne Netzwerkanbindung, wird generell nur auf die Kopie im Cache zugegriffen.

Die LINBO-Daten (Verzeichnis LINBO) können wahlweise per NFS oder SMB/SAMBA ohne Zugangspasswort read-only vom Server angeboten werden. Server-IP und NFS-Verzeichnis bzw. Sharename werden wie unter 2.2.2 auf Seite 16 angegeben, als Bootoptionen an den LINBO-Client übergeben.

Beispiel in `/etc/exports` auf dem Server für die Konfiguration per NFS (LINBO-Dateisystem Verzeichnis existiert unter `/var/linbo`):

```
/var/linbo *(ro,fsid=1000,no_root_squash,async,no_subtree_check,insecure)
```

Das Verzeichnis, welches die LINBO-Daten enthält, wird nach Start des NFS-Servers mit `/etc/init.d/nfs-kernel-server restart` in diesem Beispiel unter der Adresse

`server-ip:/var/linbo`

read-only freigegeben.

Unter SAMBA würde eine entsprechende Konfiguration in `/etc/samba/smb.conf` entsprechend eingetragen:

```
[LINBO]
comment = LINBO System
read only = yes
locking = no
path = /var/linbo
guest ok = yes
public = yes
browsable = yes
```

Hiermit wird das LINBO-Dateisystem, wieder read-only, nach Start des Samba-Servers mit `/etc/init.d/samba restart` an die Clients unter der SMB-Adresse

//server-ip/LINBO

freigegeben.

Bezüglich der Dateirechte muss darauf geachtet werden, dass die von den Clients per rsync herunterzuladenden Dateien lesbar, und die hochzuladenden Dateien schreib- bzw. überschreibbar sind. Hinweise zur Konfiguration des rsync- und SAMBA-Servers, die Probleme mit Dateirechten umgehen, sind im Abschnitt 4.1.9 auf Seite 50 zu finden.

2.1.4 LINBO-eigene Daten und Dateien

Die folgenden Dateien gehören zum LINBO Softwaresystem, und sollten für alle Clients über die zuvor angegebenen Server-Dienste erreichbar und mindestens *lesbar* sein:

Ordner/Datei	Ort	Status	Bedeutung
pxelinux.0	/tftpboot	ro	Pxelinux-Bootloader für PXE-Boot (ebenso Konfigurationsdateien unter pxelinux.cfg/HHHHHHHHH bzw. pxelinux.cfg/default
linux linux64	/tftpboot und rsync-Verzeichnis	ro	Linux-Kernel (32 bzw 64bit)
minirt.gz	/tftpboot und Rsync-Verzeichnis	ro	Initiale Ramdisk mit den Initialboot-Programmen
grub.exe	Rsync-Verzeichnis	ro	Grub4DOS-Bootloader, erforderlich für lokale Festplatteninstallation von Linux, Windows und LINBO
LINBO/LINBO	Rsync-Verzeichnis + NFS/SMB Freigabe	ro	Komprimiertes LINBO-Hauptdateisystem
LINBO/LINBO1	Rsync-Verzeichnis + NFS/SMB-Freigabe	ro	Addons-Overlay: proprietäre Komponenten (z.B. virtualbox extensions), „private“ (schulspezifische) verwendete Daten und Modifikationen, die nicht Bestandteil des Open Source LINBO Paketes sind
*.info	Rsync-Verzeichnis	ro	Textdatei mit Datumsstempel, Dateigrößen bzw. Checksummen zum

Ordner/Datei	Ort	Status	Bedeutung
			schnelleren Erkennen der Aktualität vor Download
*.cloop	Rsync-Verzeichnis	ro + rw	Komprimierte Images für native OS-Installation/Sync
VMName/*	Rsync-Verzeichnis	ro + rw	VirtualBox-VM Verzeichnis mit Konfiguration und virt. Disk-Images
start.conf start.conf-ip	Rsync-Verzeichnis und NFS/SMB-Share	ro	Dateisystem- und Startkonfiguration für Clients
logo.lss16	/tftpboot und Rsync-Verzeichnis	ro	Startbild im lss16 syslinux Format (optional)

Weitere Datei-Informationen sind im Abschnitt Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden auf Seite Fehler: Referenz nicht gefunden zu finden.

2.2 LINBO-Clients

LINBO wird üblicherweise per PXE gebootet, und kann sich selbst bootfähig auf die Cache-Partition (1.3.1) kopieren. Dadurch kann LINBO nach der ersten Installation auch dann direkt von Festplatte gestartet werden, wenn kein Netz vorhanden ist. Hierbei werden mindestens der 32-bit, der 64-bit Kernel, eine initiale Ramdisk und das komprimierte LINBO-Dateisystem /LINBO/LINBO kopiert. Der Großteil der Installationsaufgabe auf der Client-Seite besteht lediglich in der individuellen Anpassung von Konfigurationsdateien, die als Textdateien auf dem Server liegen.

2.2.1 Bootvorgang

LINBO kann wie ein normaler Linux-Kernel gebootet werden, von lokaler Festplattenpartition, bootfähigem CD-Rom, USB Flashdisk oder über das Netzwerk von einem PXE/BOOTP-fähigen DHCP-Server. Grundsätzlich ist der LINBO-Kernel auch über EFI-Systeme bootfähig, allerdings wird in den bisherigen Installationen noch kein Gebrauch von dieser Möglichkeit gemacht, zumal die unter Windows 8 übliche Standardeinstellung des Rechner-Firmware oft eine vom Hersteller kontrollierte elektronische Signatur erfordert („Secure Boot“). Das Bootschema wurde bereits in Abschnitt 1.1 auf Seite 4 behandelt.

2.2.2 LINBO- und Dateisystem-Konfiguration

Der LINBO-Client erhält seine Netzwerk-Konfigurationsdaten über Bootparameter (Kernel „append“-Zeile), die sich z.B. über die Konfigurationsdatei des Bootloaders setzen lassen, diese sind:

Parameter	Bedeutung
netdev=eth0 eth1 eth2 ...	Netzwerkkarte festlegen
hostname=rechnername	Hostname dieses Clients festlegen
ip=ip-adresse	FESTE IP-Adresse (wenn gewünscht) für diesen Client
nm=netmask	Die Netzmaske für diesen Client
gw=gateway	Das Default-Gateway für diesen Client
server=ip-adresse	Der rsync-Server für Konfigurationsdatei start.conf und Images
cache=/dev/partitionsname	Lokale Cache-Partition (analog cache=Option in start.conf)
nfsdir=nfs://ip-adresse/verzeichnis nfsdir=ip-adresse:verzeichnis	NFS-Freigabe (read-only) für die initiale Einbindung des LINBO-Dateisystems, Fallback für start.conf und Images
smbdir=smb://ip-adresse/share smbdir=//ip-adresse/share smbdir=ip-adresse:share	SMB-Freigabe (read-only) für die initiale Einbindung des LINBO-Dateisystems, Fallback für start.conf und Images
debug	Startet auf dem Client eine Debug-Shell vor dem GUI, um Fehlern auf die Spur zu kommen, oder manuell Einfluss auf die Konfiguration oder Partitionierung zu nehmen.
master	„Master-Modus“, diese Option verhindert, dass automatisch nach start.conf-Vorgaben partitioniert wird, so dass nach manuellem Einbinden einer Cache-Partition (lokal oder remote) Images vom IST-Zustand erzeugt oder eine passende start.conf angelegt werden kann.
nocache	Verhindert das automatische Einbinden der Cache-Partition (z.B. zu Testzwecken oder manuellem Umpartitionieren)

Stardmäßig bezieht LINBO die Netzwerk-Konfigurationsdaten und seinen Hostnamen per DHCP über LAN. Wird weder nfsdir noch sbmdir in der APPEND-Zeile angegeben, so ist nur der Start von einer bereits vorhandenen Installation möglich (alle vorhandenen Partitionen werden durchsucht).

Aus der Datei `start.conf-IP-Adresse`, die auf dem server per rsync-Download angeboten wird, erhält LINBO weitere Informationen über das gewünschte Dateisystem-Layout. *IP-Adresse* ist die für diesen Client per Boot-Kommandozeile oder per DHCP festgelegte IP-Adresse. Hiermit kann für jeden Rechner eine spezielle Konfiguration vereinbart werden. Um Gruppen von Rechnern mit der gleichen Konfiguration zu definieren, genügt es, für mehrere `start.conf-IP-Adresse` Dateien einen Symlink auf die gleiche tatsächliche Konfigurationsdatei anzulegen. Ein Beispiel:

```
ln -s start.conf-Windows7Starter start.conf-192.168.0.2
```

Hier wird ein Link auf eine gemeinsame Konfigurationsdatei (`start.conf-Windows7Starter` in diesem Beispiel – Groß- und Kleinschreibung wird beachtet!) unter dem neuen Namen `start.conf-192.168.0.2` angelegt.

Ist keine Datei `start.conf-IP-Adresse` auf dem rsync-Server vorhanden, wird ersatzweise die Datei `start.conf` (ohne IP-Adresse) verwendet. Diese enthält quasi die Default“-Einstellung.

Kann keine `start.conf`-Datei über das Netzwerk nachgeladen werden, so wird die auf einer bereits vorhandenen Cache-Partition liegende `start.conf` unverändert verwendet.

Der Aufbau der `start.conf`-Datei ist in Abschnitt 4.1 ab Seite 41 im Detail beschrieben.

3 Anwendung von LINBO

3.1 LINBO booten

LINBO kann sowohl übers Netz per PXE als auch von einer bereits mit LINBO installierten Festplatte gestartet werden. Die Installation eines Bootservers für LINBO ist unter Abschnitt 2.1.1 auf Seite 8 beschrieben.

LINBO besteht aus einem Kernel-, Ramdisk- und einem größeren Dateisystem-Bestandteil. Kernel und Ramdisk werden vom Bootloader (pxelinux oder grub) in den Hauptspeicher geladen. Nach einer minimalen Hardwareerkennung wird das LINBO-Dateisystem von einem Server über das Netzwerk eingebunden, oder auf einer lokalen Cache-Partition gesucht und von dort eingebunden.

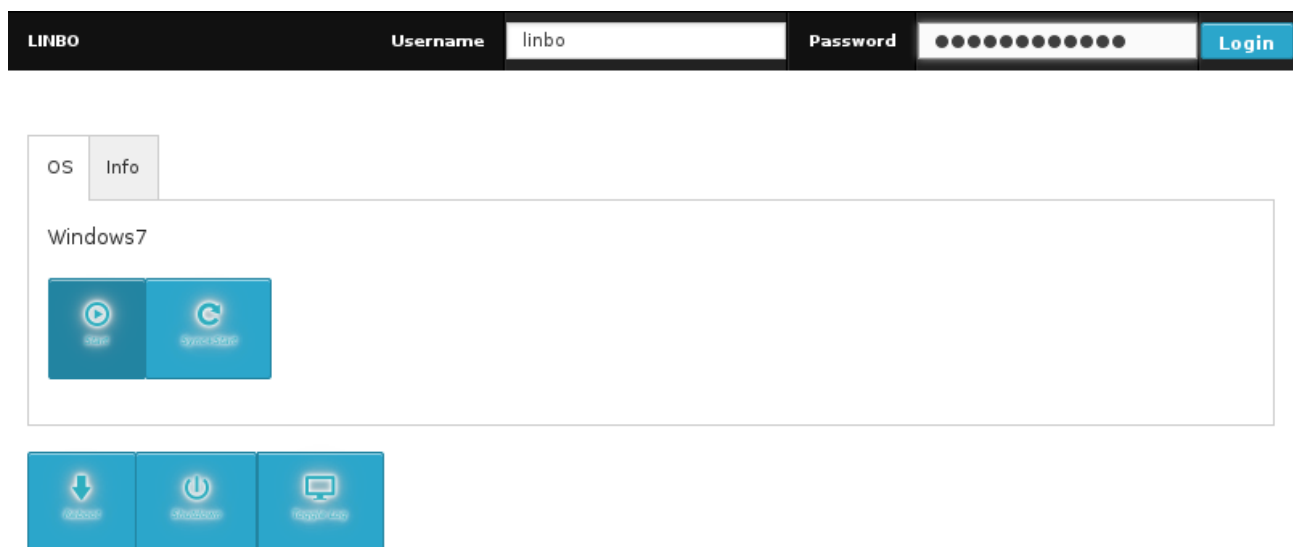
Hinweis: Die Hardwareerkennung in LINBO wurde in Version 3.0 von Mikroknoppix 7.2 übernommen, und setzt voraus, dass das komprimierte Dateisystem zur Verfügung steht, um alle Kernel-Module und Utilities

ansprechen zu können.

3.2 Graphische Oberfläche von LINBO

Nach erfolgreicher Hardwareerkennung startet der Grafikserver Xorg. Die Benutzeroberfläche ist Browserbasiert und wird durch einen Apache-Server und ein Javascript-Framework aufgrund der Informationen in `start.conf` generiert.

Während die Hauptbedienoberfläche durch einen im Fullscreen-Modus laufenden Firefox-Browser zur Verfügung gestellt wird, können einzelne Fenster und Dialoge (z.B. Konsolen-Fenster im Administrations-Modus) als separate Programme gestartet werden. Aus Gründen der Kompaktheit wurde auf eine „Desktop-Oberfläche“ mit Start-Menüs oder Kacheln verzichtet.



Fenster können durch Klick fokussiert werden. Werden Fenster vollständig durch andere verdeckt, so kann mit der Tastenkombination

Alt+Tab

zwischen den laufenden graphischen Anwendungen gewechselt, und das gewünschte Fenster wieder in den Vordergrund geholt werden. Diese Funktion ist abgeschaltet, wenn eine virtuelle Maschine aktiv ist, bzw. Alt+Tab wirkt dann nur innerhalb des virtualisierten Betriebssystems.

3.3 Betriebssysteme wiederherstellen und starten

3.3.1 Button „Start“

Mit „Start“ (Backend-Kommando `linbo_cmd start osname`) wird das ausgewählte Betriebssystem gestartet, ohne Reset oder Synchronisation, mit Ausnahme von virtuellen Maschinen, die wieder im ursprungszustand starten. Bei nativ startenden Betriebssystemen wird der zuletzt verwendete Zustand wieder gebootet.

Je nach in `start.conf` angegebener Startvariante führt der Rechner hierbei einen Reset aus (Bootreihenfolge „lokale Festplatte“ muss in diesem Fall im BIOS voreingestellt sein), oder es wird per `kexec`, wenn hardwareseitig unterstützt, ein Softboot direkt in das gewählte Betriebssystem ausgeführt.

Treten beim Starten Fehler auf, oder befindet sich das installierte Betriebssystem nach mehrfachem, unsynchronisierten Start nicht mehr in einem benutzbaren Zustand, so sollte nach dem nächsten Reboot mit Hilfe von Sync+Start wieder der zuletzt archivierte, arbeitsfähige Zustand restauriert werden.

3.3.2 Button „Sync+Start“

Mit Klick auf Sync+Start (Backend-Kommando `linbo_cmd syncstart`) wird das auf einer Partition befindliche System mit Hilfe eines auf der Cache-Partition befindlichen Archivs Datei für Datei überschrieben bzw. in den Ursprungszustand versetzt. Zuvor wird, sofern Netzzugang vorhanden und Server erreichbar, das Partitions-Archiv auf der Cachepartition vom Server per `rsync` aktualisiert.

Achtung: Hierbei gehen alle Änderungen, die in der letzten Session mit diesem Betriebssystem erstellt wurden, verloren.

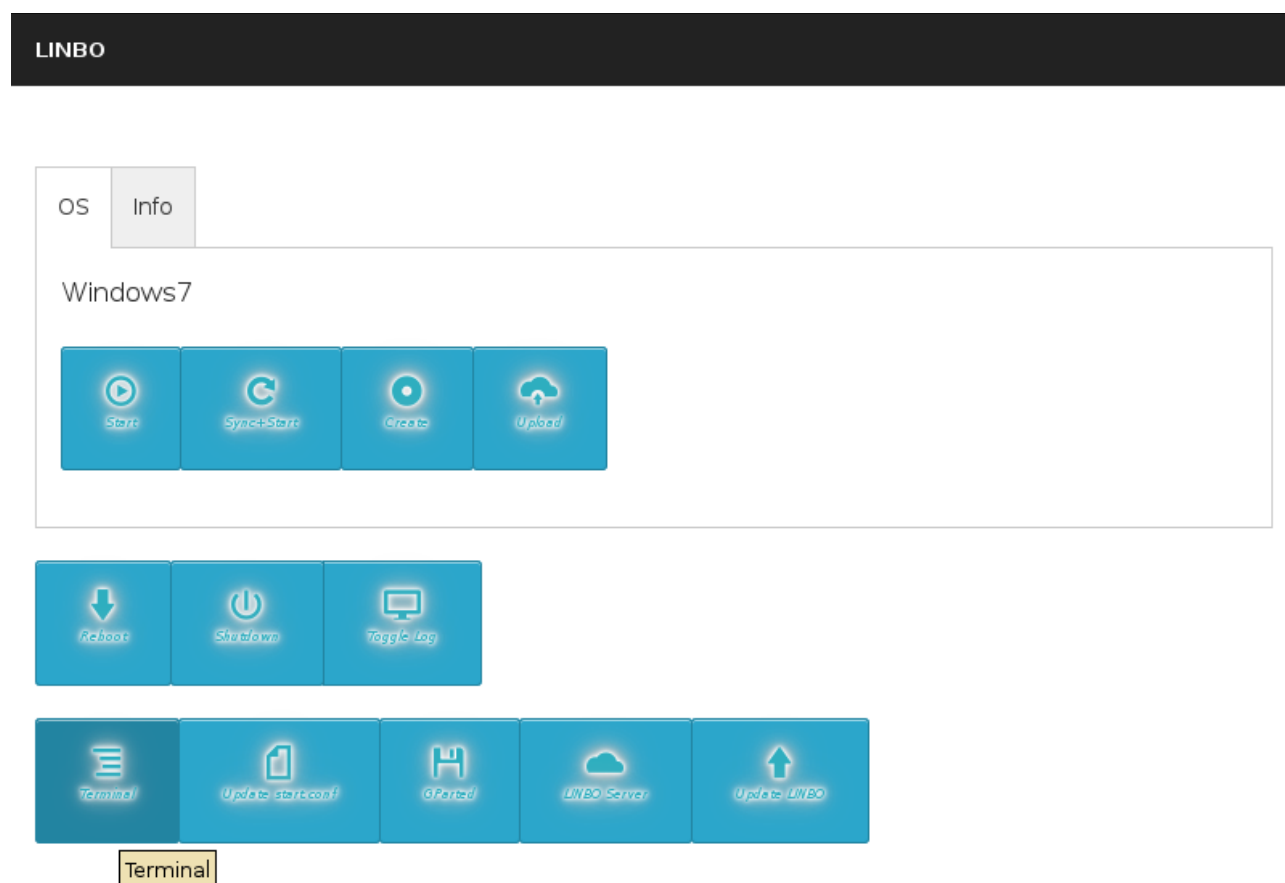
Durch Sync+Start wird ggf. eine neuere Version des jeweiligen Betriebssystems vom Server heruntergeladen und installiert. Je nachdem, ob

auf der Zielpartition bereits ein System installiert war, oder ob die Daten komplett überschrieben bzw. neu angelegt werden, nimmt der Vorgang unterschiedlich viel Zeit in Anspruch (s.a. Abschnitt Fehler: Referenz nicht gefunden Seite Fehler: Referenz nicht gefunden).

Für den anschließenden Start gelten die bereits im vorigen Abschnitt 3.3.1 erwähnten Randbedingungen.

3.4 Betriebssystem-Images erzeugen und verwalten / Admin/IT-Personal Rolle

Nach Anmeldung als LINBO-Administrator (Login und Passwort entsprechen den für die linbo-upload rsync-Freigabe festgelegten, s. Abschnitt 2.1.2 auf Seite 10) werden die Bedienelemente für administrative Funktionen in LINBO freigeschaltet. Der Benutzer wird hiermit in die Rolle des IT-Personals versetzt, und erhält Schreibzugriff auf die serverseitigen Images.



GUI Button	Funktion (Backend)	Bedeutung
„OS“ View		
Create	linbo_cmd create OSName	Erzeugt die *.clloop-Images für dieses Betriebssystem auf der Cache-Partition neu, mit dem aktuellen, zuletzt verwendeten Stand. Schreibrechte auf der Cache-Partition sind erforderlich, für den Fall, dass z.B. im Master-Mode das /cache-Verzeichnis von einem NFS/SMB-Server read/write über das Netzwerk eingebunden wurde!
Upload	linbo_cmd upload_images OSName	Lädt alle komprimierten Partitions-Images oder das VM-Verzeichnis für dieses Betriebssystem auf den Server hoch (Schreibrechte auf dem Server sind erforderlich!). Falls es durch Netzwerk-Timeouts zu einem Fehler kommt, wird der Upload noch 2 mal erneut versucht.
Modify	linbo_cmd modifyvm VMName	Startet die angegebene virtuelle Maschine <u>ohne Overlay</u> , so dass sich Modifikationen direkt auswirken (ggf. Neustart der VM bei Windows erforderlich, um Treiberinstallationen abzuschließen)
„Info“ View		
Recalculate Partition Layout		Nach Umpartitionierung, im Browser-Fenster angezeigten IST-Zustand aktualisieren.
Mount SMB		Eine SAMBA-Freigabe (bzw. Windows Fileserver) einbinden, um Images direkt auf die Freigabe zu kopieren (Schreibrechte auf dem Server notwendig)
Create/Form at Cache Partition	linbo_cmd initcache	Die Cache-Partition wird laut aktueller start.conf-Datei angelegt und formatiert. <i>Dies kann die aktuelle Partitionierung bei Abweichungen zwischen IST und SOLL-Zustand – laut start.conf - zerstören, und ist nicht für den „Master-Modus“ gedacht, in dem alle Partitionierungsaufgaben</i>

GUI Button	Funktion (Backend)	Bedeutung
		<i>manuell durchgeführt werden sollen!</i>
General		
Terminal	xterm	Startet eine Administrations-Konsole zunächst ohne root-Rechte. Hier stehen sämtliche Linux/Unix-Administrationskommandos für entsprechend Linux-erfahrenes IT-Personal zur Verfügung. Um root zu werden: <code>asroot /bin/bash</code> (normalerweise für die meisten Aufgaben nicht erforderlich)
Update start.conf	linbo_cmd update_startconf	start.conf wird erneut vom Server geladen und überschreibt ggf. lokale Modifikationen
Parted	gparted	Startet die graphische Variante des Partitionierungstools <i>parted</i> (v.a. im <i>Master-Modus</i> üblich, um manuell die Partitionierung zu ändern oder Dateisysteme zu verschieben)
LINBO Server		Öffnet ein neues Browser-Tab und zeigt die Weboberfläche des serverseitigen LINBO Konfigurations-GUI (s.a. Abschnitt 3.6)
Update LINBO	linbo_cmd update_linbo	Lädt ggf. eine neuere Version des LINBO-Bootsystems auf die Cache-Partition herunter und installiert diese bootfähig im MBR

3.5 Der LINBO „Master-Modus“

Mit der Bootoption „master“, eingegeben im Startbildschirm mit

`linbo master`

bzw. bei 64bit Systemen

`linbo64 master`

startet LINBO abweichend vom üblichen Bootverfahren:

1. Die Partitionierung nach `start.conf` wird weder überprüft, noch auf die Festplatte angewendet.
2. Die `start.conf`-Optionen `AutoPartition = yes` und `InstallMBR = yes`

werden ignoriert.

3. Es wird keine Cache-Partition eingebunden, erzeugt oder formatiert.
4. Das komprimierte LINBO-Dateisystem wird ausschließlich per NFS oder SMB eingebunden.

Die Festplatte bleibt mit dieser Option also in unverändertem Zustand (und meldet dies auch durch entsprechende Hinweise auf den „Master-Modus“ im Startvorgang). Der Rechner wird als „Master-Installation“ behandelt, alle Änderungen (inklusive Einbinden einer Cache-Partition) müssen manuell durchgeführt werden, wozu sowohl die Administrations-Konsole als auch der Partitionseditor `gparted` hilfreich sein kann.

Hinweis: Da in diesem Modus keine Cache-Partition eingebunden wird, muss vor dem Anlegen von Images ein schreibbares Verzeichnis, lokal oder remote per NFS oder SMB, nach `/cache` eingebunden werden, damit das „Create“-Kommando ausreichend Platz zum Anlegen eines komprimierten Image findet. Es ist durchaus möglich, eine auf dem Server per NFS oder SAMBA freigegebene `rsync`-Partition nach `/cache` zu mounten, und Images direkt dort abzulegen, dadurch entfällt das Hochladen.

Empfehlungen:

1. Vor dem Erzeugen eines Image sollte die lokale Kopie der `/start.conf`-Datei entsprechend modifiziert werden, so dass ein neuer Name für das Image verwendet wird, um keine bestehenden Images zu überschreiben (außer, wenn dies gewünscht ist).
2. Um das Wiederherstellen zu beschleunigen, sollten die Partitionen mit `gparted` auf eine Größe gebracht werden, die sich platzmäßig handhaben lässt. ACHTUNG: Windows benötigt ggf. einen Reboot mit anschließendem Dateisystem-Check, wenn die System-Partition verschoben oder vergrößert/verkleinert wird. Die Angaben in `start.conf`, welche für diesen Client auf dem Server vorgesehen sind, sollten auf die neuen Größen angepasst werden. Das „Info“-View im LINBO-GUI hilft dabei, die aktuellen Größen in kB festzustellen.
3. Am unproblematischsten gestaltet sich die Installation und das Imagen eines Betriebssystems, wenn die Partitionierung bereits vor der Installation des OS mit Hilfe einer entsprechend vorbereiteten `start.conf`-Datei vorgenommen wird. Dadurch werden, auch ohne Inanspruchnahme des Master-Modus, die Partitionsgrenzen richtig gesetzt und die Cache-Partition wird bereits eingerichtet, so dass das nachträgliche manuelle Einbinden kein größeres Problem mehr darstellt.

3.6 Serverseitige Konfiguration / Client-Aufnahme

Auf dem Server können über ein eigenes Webinterface `start.conf`-Dateien für Rechnergruppen vorbereitet werden, die im Zusammenspiel mit den Images für neue Clients verwendet werden können. Grundsätzlich können alle Konfigurationsdateien jedoch auch manuell, wie in diesem Handbuch exemplarisch beschrieben, angelegt und verwaltet werden. Das Webinterface ist insofern lediglich eine Arbeitserleichterung.

Der Zweck der LINBO Server-GUI ist die benutzerfreundliche Erstellung der `start.conf`, welche sowohl die Eigenschaften des LINBO Clients als auch der auf dem LINBO Client zu installierenden Betriebssysteme beschreibt.

3.6.1 Expert Configuration

Eine solche Konfiguration besteht aus der Aggregation folgender Elemente:

a) Partitionierungs-Schema (*partition selection*): das Partitionierungs-Schema beschreibt das Layout der Partitionen pro Betriebssystem, also die Größe der Partitionen, deren Device-Namen und Dateisysteme, sowie die Namen des Images je Partition.

Das Image beinhaltet die Betriebssystem-Daten der jeweiligen Partition nach der Kompression.

Beispiel: Windows-Betriebssystem mit zwei Partitionen

Expert Configuration

Please select:

Edit Partition Selections

Edit Partition Selections

Win7sys

Delete Partition Selection

Name	Device	Size	Filesystem Type	Image Name	Bootable	Description	Delete
winboot	/dev/sda1	1700000	ntfs	WinPro64_1.v	<input checked="" type="checkbox"/>	Windows 7 Pro 6	Delete
winsys	/dev/sda2	50000001	ntfs	WinPro64_2.v	<input type="checkbox"/>	Windoos 7 Pro 64	Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		

Save

Update Configuration

Beispiel: Linux-Betriebssystem mit Swap

Expert Configuration

Please select:

Edit Partition Selections

Edit Partition Selections

Linux32

Delete Partition Selection

Name	Device	Size	Filesystem Type	Image Name	Bootable	Description	Delete
root	/dev/sda1	20000000	ext4	Linux32.cloop	<input checked="" type="checkbox"/>	SolydX32	Delete
swap	/dev/sda2	2000000	swap		<input type="checkbox"/>	swap	Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		Delete
		0	ext3		<input type="checkbox"/>		

Save

Update Configuration

Die korrekte Partitionsgröße für bereits vorhandene Partitionen kann aus der LINBO Client-GUI innerhalb des Info-Bereichs abgelesen werden. Durch dieses Vorgehen kann vermieden werden, dass zu kleine Partitionsgrößen eingetragen werden.

b) Betriebssystem (operating system): ein Betriebssystem wird durch eine OsId, eine Boot Methode, eine Definition der Boot-und Root-Partition sowie im Falle eines Linux-Betriebssystems mit Boot-Parametern zum Kernel definiert. Es ist ein bereits vorhandenes Partitionierungs-Schema auszuwählen.

Konfigurations-Tipps für fortgeschrittene Anwender:

- es erleichtert die Übersicht wenn die Partitionierungs-Schemata nach den jeweiligen dafür vorgesehenen Betriebssystemen benannt werden.
- um leicht eine Versionierung von Betriebssystemen zu erreichen legt ist ein Partitionierungs-Schema mit einer Versionierung innerhalb des Image-Namens anzulegen und ein dazugehöriges Betriebssystem-Objekt, welches ebenfalls nach der gewünschten Version benannt ist.

Beispiel:

Name Partitionierungs-Schema:	Win7-v1
Image:	win7-v1.cloop
Betriebssystem:	Win7-v1

Beispiel: Betriebssystem-Definition Windows

Expert Configuration

Please select:

Edit Operating Systems ▾

Edit Operating Systems

Windows7 ▾

OSid	<input type="text" value="0"/>
Bootmethod	<input type="text" value="reboot"/>
Boot	<input type="text" value="/dev/sda1"/>
Root	<input type="text" value="/dev/sda1"/>
Kernel	<input type="text"/>
Initrd	<input type="text"/>
Append	<input type="text"/>
Patches	<input type="text"/>
Partition Selections	<input type="text" value="Win7sys"/>
Description	<input type="text" value="Windows7pro64"/>

Delete OS

Save

Update Configuration

Beispiel: Betriebssystem-Definition Linux

Expert Configuration

Please select:

Edit Operating Systems

Edit Operating Systems

Linux32

OSId	<input type="text" value="1"/>
Bootmethod	<input type="text" value="reboot"/>
Boot	<input type="text" value="/dev/sda1"/>
Root	<input type="text" value="/dev/sda1"/>
Kernel	<input type="text" value="/boot/vmlinuz-3.11-2-486"/>
Initrd	<input type="text" value="/boot/initrd.img-3.11-2-486"/>
Append	<input type="text" value="acpi_osi=Linux acpi_backl"/>
Patches	<input type="text"/>
Partition Selections	<input type="text" value="Linux32"/>
Description	<input type="text" value="Linux 32 Bit SolydX"/>

Delete OS

Save

Update Configuration

c) *Virtuelle Maschine (VM)*: eine Virtuelle Maschine wird durch einen Namen definiert. Der Name bezeichnet den Ordernamen, in welchem die virtuelle Maschine innerhalb des LINBO Caches abgelegt ist. LINBO unterstützt virtuelle Maschinen vom Typ VirtualBox.

Konfigurations-Tipps für fortgeschrittene Anwender:

- um leicht eine Versionierung von virtuellen Maschinen zu erreichen kann der VM-/Ordner-Name um ein Versions-Tag erweitert werden.

Beispiel:

Name: Win7-v1

d) *Boot-Konfiguration (PXE-Boot Configuration)*: eine Boot-Konfiguration enthält eine PXE-Boot-Konfiguration für den Client. Innerhalb der PXE-Boot Konfiguration werden notwendige Kernel-Appends definiert, welche zum Start des LINBO-Clients auf einem Rechner notwendig sind.

Beispiele:

PXE-Boot, 32 Bit-System:

```
DEFAULT linbopxe
TIMEOUT 30
# TOTALTIMEOUT 20
KBDMAP german.kbd
PROMPT 1
F1 boot.msg
F2 f2
F3 f3
DISPLAY boot.msg
SAY LINBO Boot Konsole

LABEL linbolocal
LOCALBOOT 0

LABEL linbopxe
KERNEL linux
APPEND server=10.70.1.5 smbdir=//10.70.1.5/linbo lang=de
video=vga16fb:off apm=power-off initrd=minirt.gz nomce loglevel=1
vmalloc=192M
```

PXE-Boot, 32 Bit-System im Master Mode:

```
DEFAULT linbopxe
TIMEOUT 30
# TOTALTIMEOUT 20
KBDMAP german.kbd
PROMPT 1
F1 boot.msg
F2 f2
F3 f3
DISPLAY boot.msg
SAY LINBO Boot Konsole

LABEL linbolocal
LOCALBOOT 0

LABEL linbopxe
KERNEL linux
APPEND server=10.70.1.5 smbdir=//10.70.1.5/linbo lang=de
video=vga16fb:off apm=power-off initrd=minirt.gz nomce loglevel=1
vmalloc=192M master
```

PXE-Boot, 64 Bit-System im Localboot Mode (bootet Kernel und Ramdisk aus dem Cache):

```
DEFAULT linbolocal
TIMEOUT 30
# TOTALTIMEOUT 20
KBDMAP german.kbd
PROMPT 1
F1 boot.msg
F2 f2
F3 f3
DISPLAY boot.msg
SAY LINBO Boot Konsole

LABEL linbolocal
LOCALBOOT 0

LABEL linbopxe
KERNEL linux64
APPEND server=10.70.1.5 smbdir=//10.70.1.5/linbo lang=de
video=vga16fb:off apm=power-off initrd=minirt.gz nomce loglevel=1
vmalloc=192M
```

Konfigurations-Tipps für fortgeschrittene Anwender:

- es ist ratsam, für jede Client-Klasse eine Boot-Konfiguration mit den korrekten Einstellungen für die Fälle
 - + PXE-Boot
 - + PXE-Boot Master-Mode
 - + Localbootanzulegen.

e) *Client-System (client)*: ein Client-System enthält Konfigurations-Variablen, welche die Kommunikation mit dem Server definieren (Pfad zum Samba-Verzeichnis, IP oder Hostname des Servers), eine Auswahl von vorher definierten Betriebssystemen und virtuellen Maschinen sowie eine Boot-Konfiguration. Bei Beförderung einer Client-Konfiguration zu einem Template kann diese Konfiguration im Folgenden zur einfachen Konfiguration von Clients benutzt werden (siehe *Client Wizard*).

Die Cache-Partition wird automatisch mit der maximal verfügbaren Größe angelegt. Es ist ratsam, diese als letzte Partition zu deklarieren.

Konfigurations-Tipps für fortgeschrittene Anwender:

- sofern die Konfigurationshinweise zur Versionierung von Partitions-Auswahl und Betriebssystem beachtet wurden kann einem Client nun leicht innerhalb der Dropdown-Menüs eine neue Version eines Betriebssystems zugeteilt werden.
- sollte ein Element gelöscht werden, welches von einer Client-Definition referenziert wird (etwa eine Partitionierungs-Schema), so muss ein neues Element ausgewählt werden, um die Erzeugung einer validen Konfiguration zu gewährleisten.

Beispiel:

Expert Configuration

Please select:

Edit Client Configurations

Edit Client Configurations

10.70.11.11

Boot Configuration	T40-43
Operating Systems	1 selected
Server	10.70.1.5
Cache	/dev/sda3
Smbdir	smb://10.70.1.5/LINBO
AutoStartTimeout	20
AutoStartOS	
AutoPartition	<input checked="" type="checkbox"/>
AutoSync	<input checked="" type="checkbox"/>
InstallMBR	<input checked="" type="checkbox"/>
Template	<input checked="" type="checkbox"/>
VMs	Select options
Description	IBM ThinkPad T40

Delete CLIENT

Save

Update Configuration

f) *Client-Gruppen (Client Groups)*: eine Client-Gruppe fasst eine Auswahl von Clients zusammen. Die in einer Gruppe zusammengefassten Clients werden in der Übersicht geordnet mit ihrer Beschreibung dargestellt. Die Aufnahme eines Clients in eine Client-Gruppe aktiviert für diesen Client die Erzeugung einer `start.conf-IP` und einer Boot-Konfiguration.

Die `start.conf-IP` wird auf das Server-System im konkreten Fall nach

/srv/linbo/cache/

geschrieben, die Boot-Konfiguration nach

/tftpboot/pxelinux.cfg/IP-in-HEX

Beispiel: Ansicht Übersicht mit Client-Gruppen

Overview

Client Wizard

Expert Configuration

Client Group Lenovo T410 Notebook

Client	Description

Client Group IBM ThinkPad T40-43

Client	Description

Client Group Linbo Test&Freigabe

Client	Description
10.70.11.10	T410 Win7 Pro 64Bit
10.70.11.20	T410 Win7 Pro64
10.70.11.11	IBM ThinkPad T40
10.70.8.117	Lenovo M72e VirtualBox Win7
10.70.8.240	EeePC-Testclient

3.6.2 Client Wizard

Der Client Wizard erlaubt, nachdem der fortgeschrittene Anwender innerhalb der unter 3.6.1 beschriebenen *Expert Configuration* die Systeme korrekt eingerichtet hat, die einfache Vervielfältigung einer Konfiguration für weitere Clients.

Innerhalb des GUI-Systems ist implementiert, dass Mitglieder der Django-Gruppe linboadmin das komplette Konfigurations-Interface sehen, Benutzer welche nicht Mitglied der Gruppe linboadmin sind erhalten lediglich Zugriff auf die Übersichts-Seite und den Client-Wizard.

Der Client-Wizard vereinfacht die Konfiguration eines weiteren Host-Systems auf die Auswahl einer Client-Gruppe, eines Client-Templates und die Eingabe einer IPv4-Adresse. IPv6-Adressen werden von LINBO noch nicht unterstützt.

Die Benutzerkonfiguration, welche nur für fortgeschrittene Anwender empfohlen wird, ist unter der URL

<https://linboserver/admin>

zu erreichen.

Beispiel: Ansicht Client Wizard

Quick Client Configuration

Select Client Group

Linbo-TuF

Enter IP

10.70.8.101

Select Template

T410 Win7 Pro 64Bit

Create Client

Beispiel: Ansichten Django Admin Interface

Django-Verwaltung

Benutzername:

Passwort:

Django-Verwaltung

Website-Verwaltung

Auth		
Gruppen	 Hinzufügen	 Ändern
Users	 Hinzufügen	 Ändern
Linbo server		
Client groups	 Hinzufügen	 Ändern
Clients	 Hinzufügen	 Ändern
Disks	 Hinzufügen	 Ändern
Oss	 Hinzufügen	 Ändern
Partition selections	 Hinzufügen	 Ändern
Partitions	 Hinzufügen	 Ändern
Pxlinuxcfgs	 Hinzufügen	 Ändern
Vms	 Hinzufügen	 Ändern
Sites		
Sites	 Hinzufügen	 Ändern

Kürzliche Aktionen

Meine Aktionen




-  linbo
Benutzer
-  linboadmin
Gruppe
-  testlehrer
Benutzer
-  Linux
Os
-  linbo
Benutzer
-  linbo
Benutzer

Django-Verwaltung

Start » Auth » Users

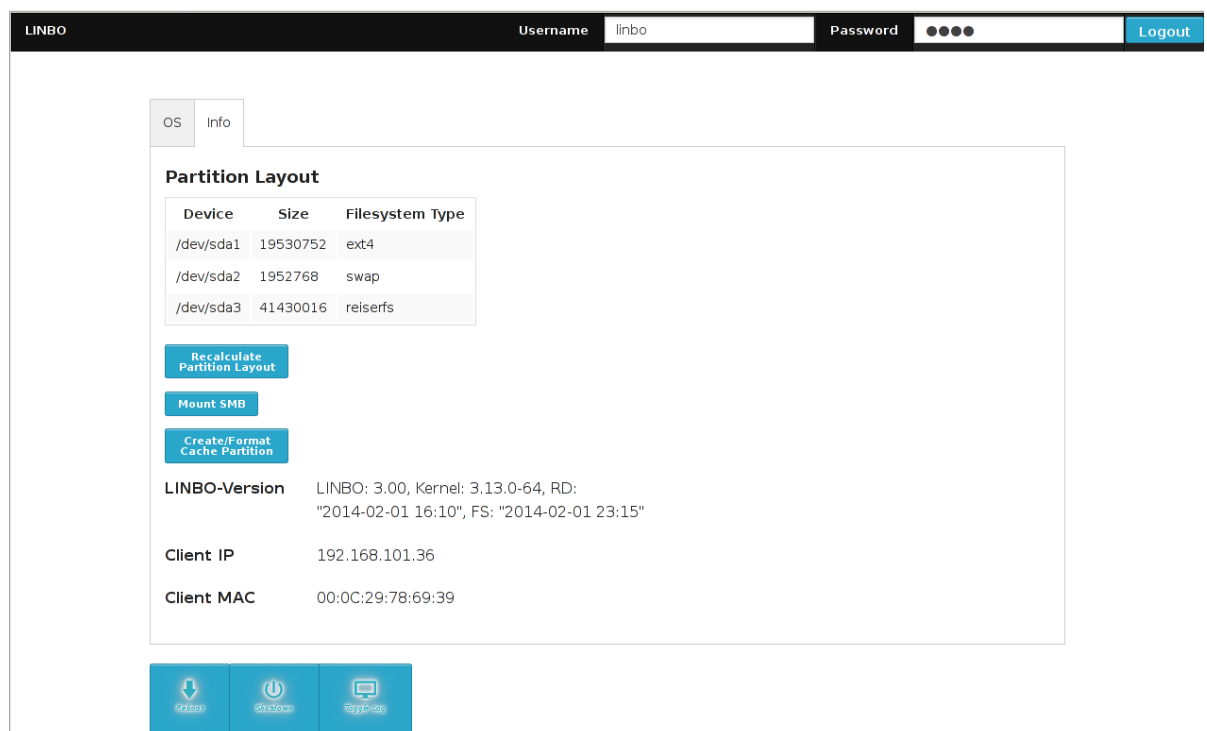
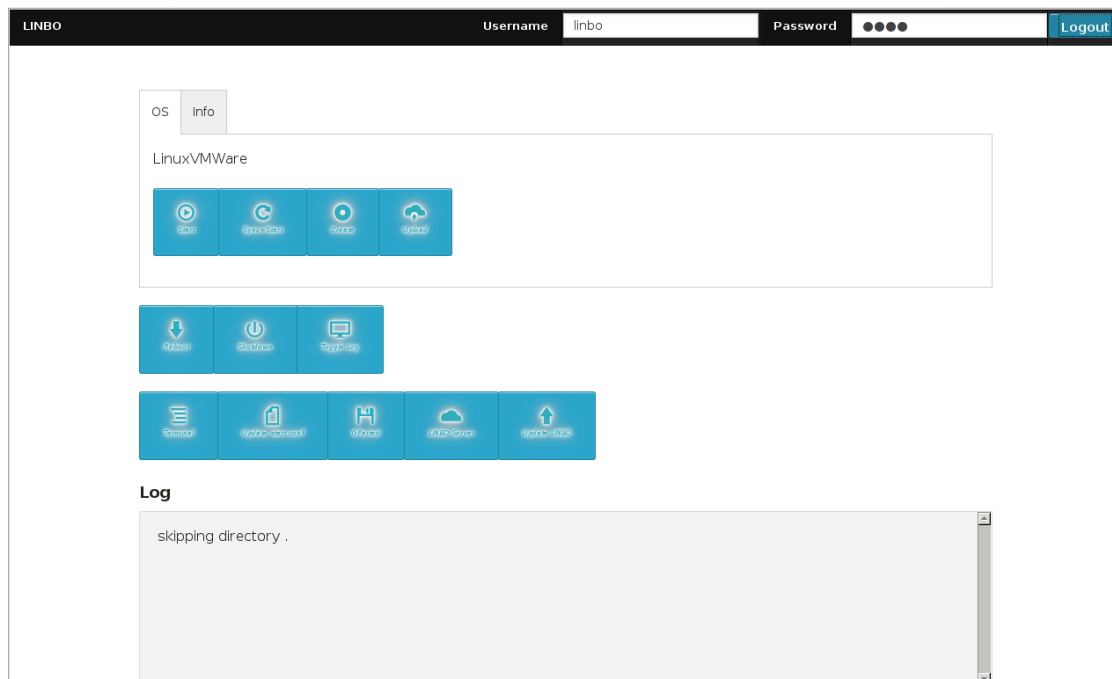
user zur Änderung auswählen

Aktion: 0 von 3 ausgewählt

<input type="checkbox"/>	Benutzername	E-Mail-Adresse	Vorname	Nachname	Redakteur-Status
<input type="checkbox"/>	linbo				
<input type="checkbox"/>	root				
<input type="checkbox"/>	testlehrer				

3 users

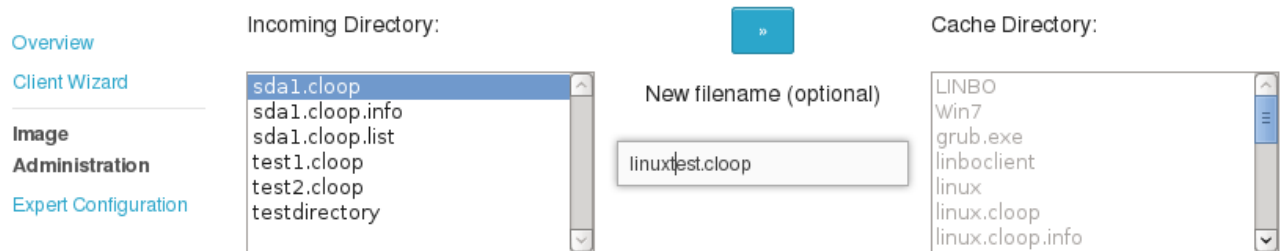
Beispiel: Resultierende Ansicht Client-GUI Hauptseite



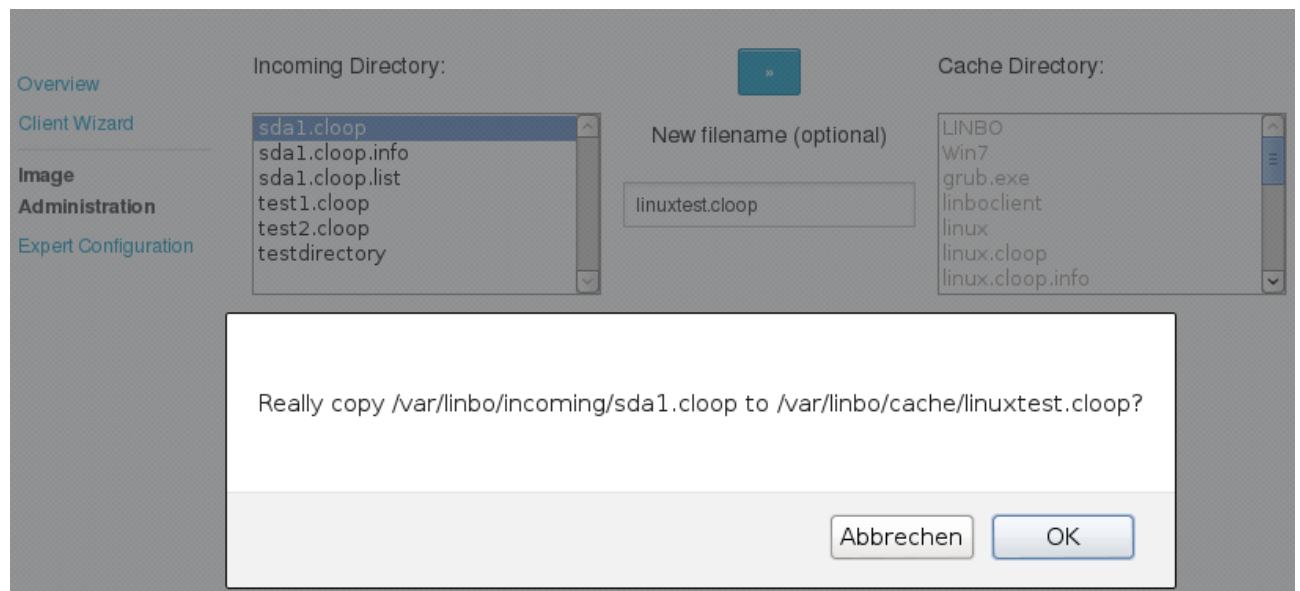
3.6.3 Image Verwaltung per GUI

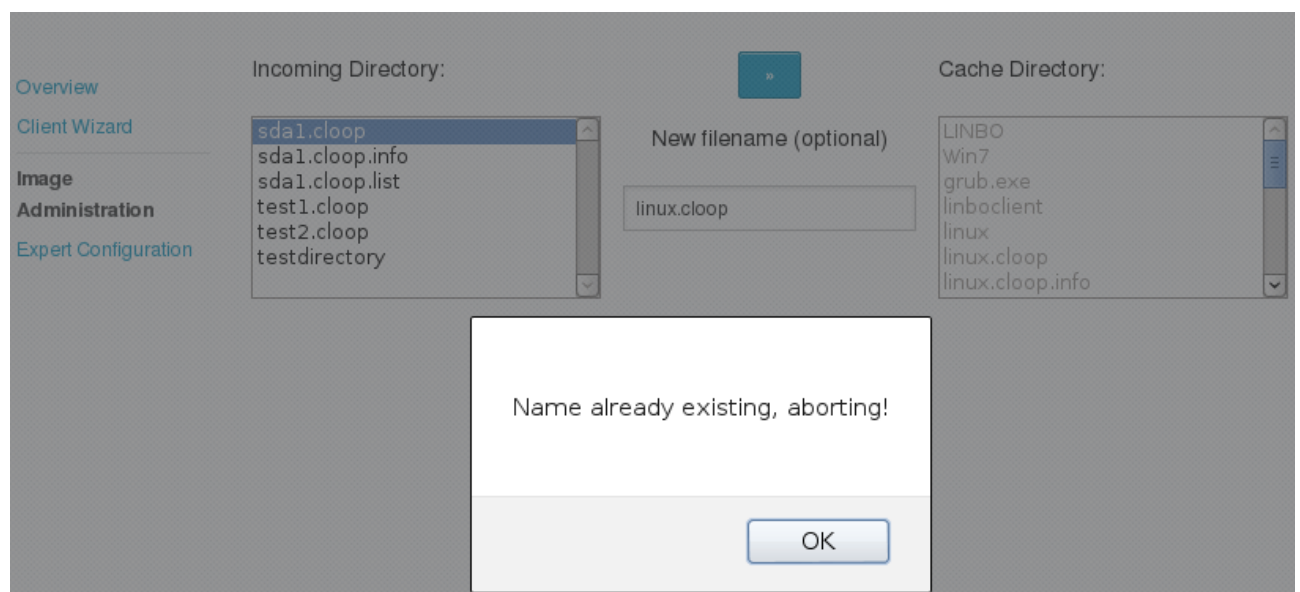
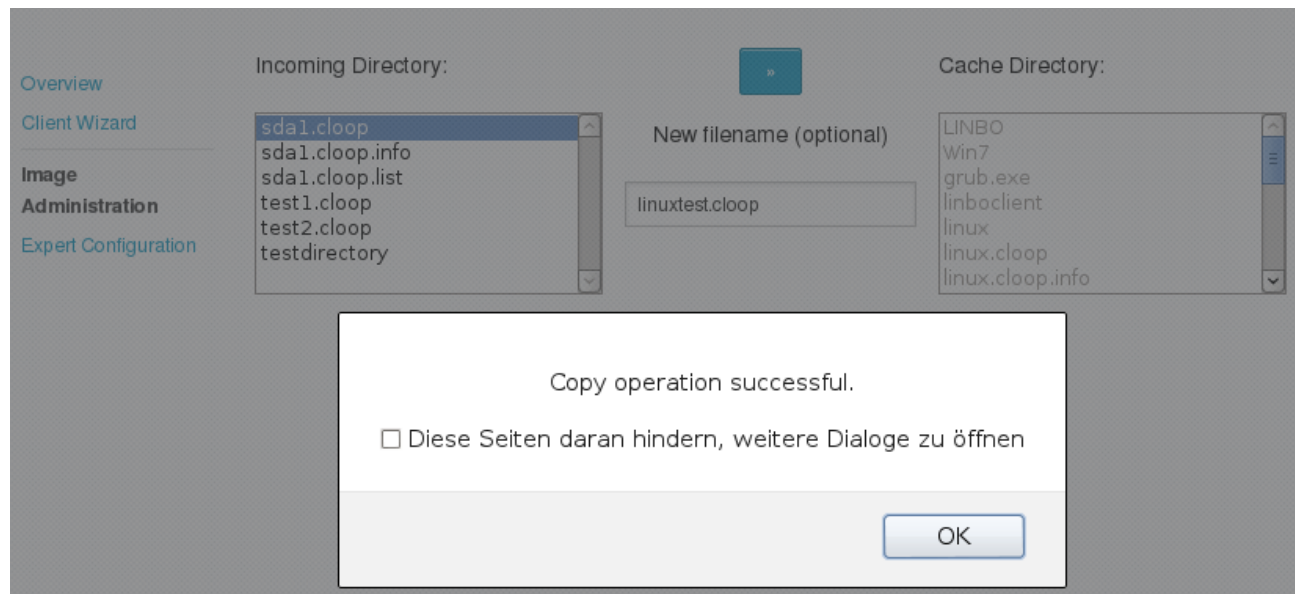
Serverseitig wurde zur Verwaltung der Images ein neues GUI-Element

integriert.



Erfolgs-, Fehlermeldungen und Rückfragen werden als „blockierender“ HTML5-Dialog dargestellt, um Fehlbedienung auszuschließen:





Entsprechend wird die Zuordnung zwischen Partitionen und Images im „Infotab“ des clientseitigen LINBO-GUI dargestellt. Es kann ein neues Image unter anderem Namen als in `start.conf` vorgesehen, erzeugt und auf den Server hochgeladen werden:

LINBO

Username

linbo

Password

●●●●

Logout

OS

Info

Partition Layout

Device	Size	Filesystem Type	Select	ImageName
/dev/sda1	2000000	ntfs	<input checked="" type="radio"/>	<input type="text" value="sda1.cloop"/>
/dev/sda2	60000001	ntfs	<input type="radio"/>	<input type="text"/>
/dev/sda3	10001	ext3	<input type="radio"/>	<input type="text"/>
/dev/sda4	42846573	reiserfs	<input type="radio"/>	<input type="text"/>
/dev/sdb1	104856600	reiserfs	<input type="radio"/>	<input type="text"/>

Recalculate Partition Layout

Mount SMB Share

Image Selected Partition

Upload Selected Partition

LINBO-Version

LINBO: 3.00, Kernel: 3.13.0-64, RD: "2014-02-28 11:50", FS: "2014-03-25 08:11"

Client IP

192.168.101.28

Client MAC

00:0C:29:FE:15:2A

4 Konfiguration von LINBO und Betriebssystemen

4.1 *start.conf* - Partitionen und Images

Die Konfigurationsdatei *start.conf* bzw. *start.conf-ipadresse* ist im Stil der bekannten KDE-Desktop-Iconbeschreibungen (ähnlich Windows *.ini*-Dateien) verfasst. Sie befindet sich im gleichen Verzeichnis auf dem Rsync-Server, in dem auch die Betriebssystem-Images für LINBO untergebracht sind.

Syntax:

Kommentare innerhalb *start.conf* werden durch **#** eingeleitet, dürfen am Anfang einer Zeile oder mitten im Text auftauchen, und werden inklusive dem bis zum Zeilenende folgendem Text vom Parser ignoriert.

Die Datei ist in Abschnitte und Optionen für diese Abschnitte eingeteilt. Abschnitte werden mit eckigen Klammern

[Abschnitt]

gekennzeichnet, Optionen haben die Form

Schlüssel = Wert

wobei die Leerzeichen jeweils vor und hinter dem Gleichheitszeichen optional, aber für die bessere Lesbarkeit empfohlen sind.

4.1.1 Abschnitte in *start.conf*- Übersicht

Abschnittsname	Bedeutung
[LINBO]	Abschnitt mit allgemeinen Einstellungen zu LINBO
[Partition]	Abschnitt mit der Definition einer Partition (Device, Dateisystem, Größe etc.)
[OS]	Abschnitt mit der Definition eines nativ zu startenden Betriebssystems
[VM]	Abschnitt mit der Definition eines virtualisiert gestarteten Betriebssystems

Abschnittsname	Bedeutung
[Disk]	Abschnitt mit der Angabe des Partitionierungsschemas einer Festplatte

4.1.2 Abschnitt [LINBO]

Schlüssel	Mögl. Werte	Bedeutung
Cache	Partitionsname, z.B. /dev/sda3	Partition, die unter /cache schreibbar eingebunden wird, und die die lokalen Kopien von Images für Sync+Start sowie die LINBO-Bootdateien für den lokalen Start enthält
Server	IP-Adresse, z.B. 10.70.1.5	Rsync-Server IP-Adresse mit linbo / linbo-upload Freigaben
AutoStartTimeout	Zeit in Sekunden, z.B. 15	Zeitspanne, nach der bei Inaktivität ein Betriebssystem vom GUI automatisch gestartet wird
AutoStartOS	OS oder VM Name	Name des nativen oder virtualisierten Betriebssystems aus den Abschnitten [OS] oder [VM], das automatisch gestartet werden soll
AutoPartition	yes oder no	Soll bereits beim Hochfahren ohne Rückfrage die Partitionierung nach start.conf auf dem Client (wieder-)hergestellt werden?
AutoSync	yes oder no	Sollen bereits beim Hochfahren die Images vom Server heruntergeladen und aktualisiert werden?
InstallMBR	yes oder no	Soll bereits beim Hochfahren ein Master Boot Record (MBR) zum lokalen Start von LINBO erzeugt werden?
TorrentEnabled	yes oder no	Sollen (für die optionale Download-Methode torrent) für jedes Image Tracker gestartet und entsprechende Synchronisationsdateien angelegt, und der zentrale Server über vorhandene Images informiert werden?

4.1.3 Abschnitt [Disk]

Schlüssel	Mögl. Werte	Bedeutung
Dev	Festplatten-Devicename, z.B. /dev/sda	Name/Adresse der Disk
Type	msdos oder gpt	Art der Partitionstabelle für die automatische Partitionierung per parted. msdos ist die „traditionelle“ Partitionstabelle mit 4 primären und beliebig vielen logischen Partitionen, gpt die neuere, seltener verwendete Partitionierung, die mitunter bei Festplatten größer 2 Terabyte zum Einsatz kommt.

4.1.4 Abschnitt [Partition]

Schlüssel	Mögl. Werte	Bedeutung
Dev	Partitions-Devicename, z.B. /dev/sda1	Device/Adresse der Partition, auf die sich die weiteren Angaben beziehen
OsId	Nummer	Zuordnung zu einem nativ installierten Betriebssystem (nur Partitionen, die zu einem OS gehören, werden bei Sync+Start berücksichtigt)
Size	Positive Zahl oder -1	Angabe der Größe in KiloByte (kB), die für diese Partition vorgeschrieben ist. Der Wert -1 bedeutet „restlicher verfügbarer Platz“, und darf pro primärem bzw. logischem Partitionstyp nur einmal verwendet werden.
FStype	swap, ext2, ext3, ext4, reiserfs, ntfs, fat32, <i>extended</i>	Dateisystemtyp für die Formatierung der Partition. Bei <i>extended</i> wird eine sog. „erweiterte“ Partition angelegt (nur bei den primären Partitionen /dev/sd?[1-4] möglich!), auf der dann weitere logische Partitionen (/dev/sd?5 ff) angelegt werden können)
Image	<i>datei.cloop</i>	Name der Image-Datei, die auf dieser Partition zu installieren bzw. zu synchronisieren ist.
Bootable	yes oder no	Soll das „Bootable“-Flag in der Partitionstabelle für diese Partition

Schlüssel	Mögl. Werte	Bedeutung
		aktiviert werden? (Ggf. für ältere Rechner notwendig, deren BIOS sonst die Partition nicht als bootfähig erkennt)
Quicksync	Verzeichnis 1, Verzeichnis 2, ...	Komma-getrennte Liste von Verzeichnissen (dürfen Leerzeichen enthalten, dafür keine Leerzeichen direkt vor und hinter dem Komma). Ist dieser Parameter angegeben, so werden NUR die angegebenen Verzeichnisse auf dieser Partition synchronisiert, um Zeit zu sparen. Fehlt eines der angegebenen Verzeichnisse auf der Partition, so findet eine vollständige Synchronisation statt, so wie es auch ohne die Option Voreinstellung ist.

4.1.5 Abschnitt [OS]

Schlüssel	Mögl. Werte	Bedeutung
OsId	Nummer	S.a. Abschnitt [Partition]: Zuordnung zu einem nativ installierten Betriebssystem (nur Partitionen, die zu einem OS gehören, werden bei Sync+Start berücksichtigt)
Name	Name ohne Leer- und Sonderzeichen, z.B. Windows7	Name (Kürzel) für das Betriebssystem, so wie er auch als Beschriftung im GUI auftaucht
Description	Text ...	Längere Beschreibung des Betriebssystems, Kommentare etc.
Method	kexec, reboot, local	<p>kexec: Kernel (oder Bootloader grub.exe bei Windows) und initiale Ramdisk werden im „Softboot-Verfahren“ direkt geladen und ohne Reboot gestartet. Diese Methode setzt kompatible Hardware voraus, die keinen „Hardreset“ benötigt, um sich neu zu initialisieren.</p> <p>reboot: Das Betriebssystem wird in die Konfiguration des Bootloaders grub.exe auf der Cache-Partition und im MBR so eingetragen, dass es beim nächsten lokalen Reboot einmalig gestartet wird (danach wird LINBO reaktiviert). Anschließend wird ein Hardware-Reset</p>

Schlüssel	Mögl. Werte	Bedeutung
		<p>ausgelöst, um den Rechner neu zu starten und alle Komponenten in den Ursprungszustand zu bringen. Im BIOS muss der Start von Festplatte als erster Punkt in der Bootreihenfolge aktiv sein.</p> <p>local: Wie „reboot“, allerdings wird der Bootloader direkt auf der Betriebssystem-Partition installiert, und nicht auf der Cache-Partition. Das gewählte Betriebssystem wird permanent als Standard eingetragen. Hiermit lässt sich LINBO auch als reine Imaging/Rollout-Lösung nutzen. Eine „Reparatur“ oder Synchronisation findet in diesem Modus nicht statt.</p>
Boot	Partitionsname, z.B. /dev/sda1	Bootpartition, auf der sich der Betriebssystem-Kernel und initial Ramdisk oder (bei Windows) der Bootloader, sowie die am nativen Bootvorgang des OS beteiligten Dateien befinden. Kann mit der Root-Partition identisch sein.
Root	Partitionsname, z.B. /dev/sda2	Root-Partition, auf der sich i.d.R. die für den weiteren Bootvorgang notwendigen Programme und Dateien des Betriebssystems befinden. Kann mit der Boot-Partition identisch sein.
Kernel	Pfad/Dateiname, z.B. /boot/vmlinuz-3.13.0	Pfad zum Systemkernel auf der Boot-Partition, alternativ „grub.exe“, wenn über den Umweg des grub4dos-Bootloaders gestartet wird, um Grafikprobleme zu umgehen.
Initrd	Pfad/Dateiname, z.B. /boot/initrd-3.13.0	Pfad zu einer initialen Ramdisk (Linux-spezifisch) auf der Boot-Partition.
Append	Liste mit Optionen, durch Leerzeichen getrennt	Kernel- oder Bootloader-spezifische Optionen, die beim Starten mitgegeben werden. Im Falle von grub.exe als Kernel sind dies Konfigurationsoptionen für den grub-Bootloader, die auch den Pfad zum Linux-Kernel bzw. Windows-Bootloader in der grub-typischen Syntax enthalten

Schlüssel	Mögl. Werte	Bedeutung
Patches	Dateiname(n), z.B. Windows7.reg	Leerzeichen-separierte Liste von Patch-Dateien, die auf das Betriebssystem angewendet werden, um kleinere Modifikationen vorzunehmen. Für Windows typisch sind .reg-Dateien im regedit-Format, die Änderungen an der Windows-Registry enthalten, z.B. um den Hostnamen des Client gemäß der aktuellen IP-Adresse zu setzen oder spezielle Systemeigenschaften zu ändern. Für Linux-Systeme sind normalerweise keine Patches notwendig.

4.1.6 Abschnitt [VM]

Schlüssel	Mögl. Werte	Bedeutung
Name	Name ohne Leer- und Sonderzeichen, z.B. Windows7	Name (Kürzel) für das virtualisierte Betriebssystem, so wie er auch als Beschriftung im GUI auftaucht. Muss mit dem Namen des VM-Verzeichnisses identisch sein.
Patches	Dateiname(n), z.B. Windows7.reg	Leerzeichen-separierte Liste von Patch-Dateien, die auf das virtualisierte Betriebssystem angewendet werden, um kleinere Modifikationen vorzunehmen. Für Windows typisch sind .reg-Dateien im regedit-Format, die Änderungen an der Windows-Registry enthalten, z.B. um den Hostnamen des Client gemäß der aktuellen IP-Adresse zu setzen oder spezielle Systemeigenschaften zu ändern. Für Linux-Systeme sind normalerweise keine Patches notwendig. Die Änderung wird direkt in der virtuellen Festplatte (.vdi-Datei) des Betriebssystems durchgeführt, daher ist es empfehlenswert, eine Name.info-Datei mit Zeitstempel beizulegen, die sich durch das Patchen nicht ändert, da sonst die Änderung der i.d.R. sehr großen VDI-Datei von rsync gegenüber der unveränderten Version auf dem Server erkannt und die Daten bei jedem remote-Sync neu übertragen werden!

4.1.7 Beispiel für eine gültige start.conf-Datei

```
# Aufteilung (gesamt 80GB):
# /dev/sda1 Windows boot (2GB)
# /dev/sda2 Windows system (50GB)
# /dev/sda3 extended (Rest)
# /dev/sda5 Linux (5GB)
# /dev/sda6 Linux-Swap (1GB)
# /dev/sda7 Cache (Rest)

[LINBO]
Cache = /dev/sda7          # Name der Cache-Partition
Server = 192.168.100.254   # IP-Adresse des Rsync-Servers
AutoPartition = yes        # Automatisch partitionieren
AutoSync = yes             # Automatisch Images synchronisieren
InstallMBR = yes           # LINBO im Cache bootfähig installieren
TorrentEnabled = no        # Keinen Torrent-Server starten

[Disk]
Dev = /dev/sda             # erste SATA/SCSI/SSD Disk
Type = msdos               # msdos or gpt

[Partition]
Dev = /dev/sda7            # 7. Partition
Size = -1                  # Rest des verfügbaren Platzes
FSType = reiserfs          # Reiserfs-Dateisystem (autoreparierend)
Bootable = no              # kein "Bootable"-Flag

[Partition]
Dev = /dev/sda3            # 3. Partition
Size = -1                  # Rest des verfügbaren Platzes
FSType = extended          # Erweiterte Partition
Bootable = no              # kein "Bootable"-Flag

[Partition]
Osid = 1                   # Gehört zum OS Nr. 1
Dev = /dev/sda1            # 1. Partition
Size = 2000000             # 2GB
Image = windows1.cloop     # komprimierte Image-Datei
FSType = ntfs              # NTFS-Dateisystem (Windows)
Bootable = yes             # Partition als bootfähig markieren

[Partition]
Osid = 1                   # gehört ebenfalls zu OS Nr. 1
Dev = /dev/sda2            # 2. Partition
Size = 50000000            # 50GB
Image = windows2.cloop     # komprimierte Image-Datei
```

```
FSType = ntfs          # NTFS-Dateisystem (Windows)
Bootable = yes         # Partition als bootfähig markieren
```

```
[Partition]
Osid = 2               # gehört zu OS Nr. 2
Dev = /dev/sda5        # erste logische Partition
Size = 5000000         # 5GB
Image = linux.cloop    # komprimierte Image-Datei
FSType = ext4          # ext4-Dateisystem (Linux)
Bootable = no         # kein "Bootable"-Flag
```

```
[Partition]
Osid = 2               # gehört ebenfalls zu OS Nr. 2
Dev = /dev/sda6        # zweite logische Partition
Size = 1000000         # 1GB
FSType = swap          # Swap-Signatur anlegen
Bootable = no         # kein "Bootable"-Flag
```

```
[OS]
Osid = 2               # Dies ist OS Nr. 2
Name = Linux           # Kürzel/Name
Version = 1            # Zusatzinfo, ignoriert
Description = Ubuntu   # Beschreibung
Boot = /dev/sda5       # Boot-Partition
Root = /dev/sda5       # Root-Partition (die gleiche)
Kernel = vmlinuz-3.2.0-4-amd64 # Kernel
Initrd = initrd.img-3.2.0-4-amd64 # Initiale Ramdisk
Append = nosplash vga=792 # Kernel-Optionen
Patches =              # Keine Patches
```

```
[OS]
Osid = 1               # Dies ist OS Nr. 1
Name = Windows         # Kürzel/Name
Version = 1            # Zusatzinfo, ignoriert
Description = ohne Viren # Beschreibung
Boot = /dev/sda1       # Boot-Partition mit Windows-Bootloader
Root = /dev/sda2       # Root (Windows System) Partition
Kernel = reboot        # Bootmethode mit Hardreset
Method = reboot        # Bootmethode mit Hardreset (alternativ)
Initrd =               # keine initiale Ramdisk
Append =               # keine Bootoptionen
Patches = windows1.cloop.reg # Registry-Patch
```

```
[VM]
Name = Win7            # Name/Kürzel/Verzeichnisname
Patches =              # Keine Patches
```


Sofern Controller und Board den kexec-Softboot unterstützen („Method = kexec“), kann Windows auch per „Kernel = grub.exe“ gestartet werden. In diesem Fall wären geeignete Optionen für Append:

```
Append = --config-file=chainloader (hd0,0)/bootmgr
```

um den Windows-Bootlader bootmgr per grub direkt zu starten.

4.1.8 Windows-Patches - .reg-Dateien

Vor dem nativen Start von Windows kann in die auf der Partition installierte Kopie ein Registry-Patch integriert werden. Hierzu wird eine modifizierte Variante von chntpw verwendet, die im Gegensatz zum Original auch dazu in der Lage ist, das *Hive*-Format, in dem die Windows-Registry abgelegt ist, durch neue Einträge zu erweitern.

Das Format der .reg-Patch-Dateien ähnelt dem der in der Windows-Systemadministration bekannten regedit-Dateien, die in einem einfachen Textformat abgelegt sind:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ComputerName\ActiveComputerName\]
"ComputerName"="{${HostName$}}"

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ComputerName\ComputerName\]
"ComputerName"="{${HostName$}}"

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\]
"Hostname"="{${HostName$}}"
"NV Hostname"="{${HostName$}}"

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System\]
"DefaultLogonDomain"="WORKGROUP"
```

Die Variable `{${HostName$}}` in diesem Beispiel wird durch den Hostnamen ersetzt, den LINBO per DHCP oder per IP-Adressen Auflösung über DNS erhält. Eindeutige Hostnamen sind für Windows für die Anbindung des Rechners an eine Domäne wichtig, daher ist es sinnvoll, den Hostnamen korrekt in die Registry an den für die jeweilige Windows-Version vorgesehenen Stellen der Registry einzupatchen.

Die angegebenen Patch-Dateien, für die die Endung .reg vorgeschrieben ist, werden an der gleichen Stelle abgelegt wie die cloop-Images, die Partitionen

für die Betriebssysteme enthalten. Bei virtuellen Maschinen befindet sich die zugehörige Patch-Datei auf der gleichen Verzeichnisebene wie das Verzeichnis, das die .vdi-Dateien enthält (der Patch liegt also NICHT im VM-Verzeichnis, sondern außerhalb).

Um für einen Client rein lokale Registry-Einträge (wie sie z.B. nach einem Domain-Join auftreten) abzubilden, wird zusätzlich eine Datei namens

OSNAME-local.reg

auf der Cache-Partition, falls vorhanden, in die Registry des per *OSNAME* angegebenen Betriebssystems (Schlüsselwort Name in start.conf) eingepatcht. Hierzu ist kein weiterer Eintrag in start.conf erforderlich.

Für Linux-Betriebssysteme sind keine Patch-Dateien notwendig, hier wird der Hostname kurz vor dem Start des nativen oder virtualisierten Systems in /etc/hostname, die Root-Partition in /etc/fstab eingetragen.

4.1.9 Upload-Dateirechte automatisch korrigieren

Beim Upload per rsync oder über einen SAMBA-Server kann es zu Problemen mit den Dateirechten der hochgeladenen Dateien kommen, wenn der unprivilegierte rsync Download Betriebssystem-Images, welche mit restriktiveren Rechten abgelegt sind, nicht lesen kann.

Durch Hinzufügen von Konfigurationsdirektiven in der Konfiguration von rsync oder SAMBA lassen sich die Dateirechte nach Hochladen automatisch korrigieren:

/etc/rsyncd.conf (Abschnitt [linbo-upload]):

```
incoming chmod = Dugo=rwx,Fugo=rw
```

/etc/samba/smb.conf (Abschnitt des Upload-Verzeichnisses):

```
force create mode = 0666  
force directory mode = 0777
```

Weiterhin kann per cron-job eine zeitgesteuerte Korrektur der Dateirechte des LINBO Upload-/Download-Verzeichnisses durchgeführt werden, indem z.B. alle 5 Minuten das Kommando

```
chmod -R a+rwX /Pfad/zum/Linbo-Verzeichnis
```

ausgeführt wird.

5 Das LINBO-Kochbuch, „How do I ...?“

5.1 Der allererste Start: Wie richte ich ein Master-System für die Erzeugung von LINBO-Images ein?

Grundsätzlich ist es praktisch, wenn das Master-System auch schon mit Hilfe von LINBO partitioniert wird, damit die Partitionsgrößen und -namen zuverlässig festgelegt sind.

```
[LINBO]
Cache = /dev/sda2
Server = 10.0.2.2
```

```
[Partition]
Osid = 1
Dev = /dev/sda1          # Windows-Partition
Size = 40000000          # 40GB
FSType = ntfs
Bootable = yes
Image = win.cloop
```

```
[Partition]
Dev = /dev/sda2          # Cache-Partition
Size = 40000000          # 40GB
FSType = reiserfs
Bootable = no
```

```
[OS]
Osid = 1
Name = Win
Description = Installation vom 06.11.2013
Boot = /dev/sda1
Root = /dev/sda1
Kernel = grub.exe
Initrd =
Method = kexec
Append = --config-file=chainloader (hd0,0)/bootmgr; boot
```

Mit einer minimalen start.conf-Datei wie der hier gezeigten könnte begonnen werden. Nach der Partitionierung mit Anlegen von System- und Cache-Partition kann die Windows-Installation vom Installationsmedium durchgeführt werden. Bitte beachten Sie, dass für LINBO eine Cache-Partition (hier: /dev/sda2) eingerichtet werden muss, die groß genug ist, um alle Betriebssysteme vorzuhalten, die auf den Clients installiert werden sollen (plus etwas Platz für LINBO selbst).

5.2 Wie groß soll die Cache-Partition sein, und wo genau soll sie auf der Festplatte liegen?

Die Cache-Partition hält (außer bei der Bootmethode „local“) eine Kopie der Installations-Images für jedes Betriebssystem, das auf den Clients bei Bedarf neu aufgesetzt, synchronisiert oder aktualisiert werden soll. Grundsätzlich ist jede Partitionsnummer zulässig, neben primären (sda1...sda4) auch logische Partitionen (sda5 ff).

Der Name der Partition muss in der `start.conf` unter Abschnitt `[LINB0]`, Parametername `Cache` eingetragen werden, und die Größe in einem Abschnitt `[Partition]` so wie in diesem Beispiel:

```
[LINB0]
Server = 10.0.2.2
Cache = /dev/sda2                # <- Das ist die Cache-Partition

[Partition]
Dev = /dev/sda2                  # Name
Size = 20000000                  # Größe in kB
FSType = reiserfs                 # Dateisystem
Bootable = no                    # Bootflag nicht gesetzt
```

In diesem Beispiel (die Kommentare mit `#` sind optional) wird eine 20 GB große Cache-Partition verwendet, die mit `reiserfs` formatiert ist. Generell wird für die Cache-Partition `reiserfs` empfohlen, da es sich selbst beim Einbinden ohne Dateisystemcheck selbst repariert und keine relevanten Einschränkungen in Datei- oder Partitionsgrößen hat.

5.3 Wie setzen sich die Partitionsnamen unter Linux zusammen, was muss ich angeben?

Das Standard `msdos`-Partitionsschema bei PC-Festplatten erlaubt, unabhängig davon, ob Linux oder Windows zum Einsatz kommt, maximal 4 primäre Partitionen. Unter Linux heißen diese `/dev/sda1 ... /dev/sda4` (SCSI, SATA, SSD, USB-Flash oder PATA) für die erste Festplatte bzw. `/dev/sdb1 ... /dev/sdb4` für die zweite. Bei Windows gibt es hingegen gar kein einheitliches Schema zwischen „Laufwerksbuchstaben“, wobei aber `C:` oftmals die erste primäre Festplattenpartition ist, auf der Windows installiert ist, als Devicename unter Linux entsprechend `/dev/sda1`.

Eine der primären (ersten 4) Partitionen darf eine „erweiterte“ Partition (Partitions-Kennung 5 bzw. `FSType=extended` in `start.conf`) sein, auf der sich dann weitere sog. „logische“ Partitionen einrichten lassen. Dies ist sinnvoll, wenn die Anzahl der insgesamt benötigten Partitionen größer ist als die als „primäre“ erlaubten 4 Partitionen sind. Beim auf sehr großen Festplatten (größer 2TB) verwendeten GUID Partitionsschema (`Type = gpt` im Abschnitt

[Disk] in `start.conf`, s.a. Abschnitt 4.1.3 Seite 43) existieren keine erweiterten Partitionen, und es wird nicht zwischen primären und logischen Partitionen unterschieden. Nicht jedes BIOS unterstützt jedoch das Booten von GPT-partitionierten Festplatten, ggf. kann hier eine kleinere Festplatte mit `msdos`-Schema als erste Festplatte verwendet werden, und eine größere mit `gpt`-Schema für „Datenpartitionen“.

Ein Linux-System kann auf jeder beliebigen Partition installiert werden, der Kernel erkennt auch die „logischen“ Partitionen des erweiterten Partitionsschemas, um das Wurzelverzeichnis (Root) zu mounten. Für Linux-Betriebssysteme sollte allerdings auch eine Swap-Partition vorhanden sein, die als RAM-Erweiterung dient und bei Speichermangel die gerade nicht benötigten Daten aufnehmen bzw. auslagern kann. Empfohlene Größe für die Swap-Partition ist je nach Bedarf und Größe des Hauptspeichers üblicherweise 1-4 GB, Partitionstyp ist 82 („Linux swap“, `FSType = swap` in `start.conf`).

Devicename	Bedeutung
<code>/dev/sda</code>	Erste Festplatte komplett (SCSI, SATA, SSD, PATA, USB)
<code>/dev/sda1</code>	Erste Festplatte, erste primäre Partition
<code>/dev/sda2</code>	Erste Festplatte, zweite primäre Partition
<code>/dev/sda3</code>	Erste Festplatte, dritte primäre Partition
<code>/dev/sda4</code>	Erste Festplatte, vierte primäre Partition
<code>/dev/sda5</code>	Erste Festplatte, erste „logische“ Partition (auch dann, wenn eine der vier primären Partitionen fehlt bzw. leer ist).
<code>/dev/sda6</code>	Erste Festplatte, zweite „logische“ Partition
<code>/dev/sdb</code>	Zweite Festplatte komplett (SCSI, SATA, SSD, PATA, USB)
<code>/dev/sdb1</code>	Zweite Festplatte, erste primäre Partition
<code>/dev/hda</code>	Erste Festplatte komplett (älterer IDE-Treiber, veraltet)
...	...

5.4 Welche Dateisystem-Typen muss ich in `start.conf` angeben? (`FSType = ?`)

Windows kennt kaum Linux-native Dateisysteme, und bietet mitunter im laufenden Betrieb an, die „ungenutzten Bereiche“ der Festplatte zu formatieren und somit für Windows nutzbar zu machen, was ein kundiger Windows-Administrator unterbinden sollte.

Folgende Dateisystem-Typen sind unter Linux und Windows gebräuchlich:

Dateisystem	Beschreibung
reiserfs	reiserfs merkt sich in einem „Journal“ die zuletzt durchgeführten Änderungen, und stellt nach einem Systemausfall beim Einbinden des Dateisystems den letzten konsistenten Zustand automatisch wieder her. Es ist durch Verwendung von balancierten Suchbäumen sehr schnell beim Zugriff auf viele kleine Dateien in verschachtelten Verzeichnissen, reagiert aber empfindlicher auf physikalische Festplattendefekte als ext2 oder ext3.
ext2	Linux-typisches Dateisystem, unterstützt alle für Unix-Systeme typische Dateitypen und Dateirechte.
ext3	Wie ext2 und auch kompatibel dazu, aber mit zusätzlichem Journal“ für die beschleunigte Reparatur per Filesystemcheck nach einem Crash. Im Gegensatz zu reiserfs wird ext3 beim Mounten im Fehlerfall nicht automatisch repariert, sondern muss mit fsck.ext3 vom gestarteten Betriebssystem geprüft und ggf. repariert werden. Bei Sync+Start überprüft LINBO das Dateisystem automatisch auf Fehler und korrigiert diese gegebenenfalls, sofern ohne Interaktion möglich.
ext4	Ähnlich ext3, mit erweiterten Optionen für schnelleren Dateizugriff.
fat32/vfat	Das klassische „MSDOS“-Dateisystem mit eingeschränkten Dateinamen und einer Maximalgröße für Dateien bis 4GB. Wurde bis Windows XP standardmäßig verwendet, und ist bis heute das bevorzugte Dateisystem auf USB-Sticks oder SD-Karten, da sehr einfach aufgebaut.
ntfs	„New Technology File System“, wird von Windows-Versionen ab Vista standardmäßig verwendet. Es unterstützt einige, jedoch nicht alle der von Unix/Linux her bekannten Features wie lange Dateinamen, internationale Zeichensätze in Dateinamen, Verknüpfungen, erweiterte Dateirechte und Dateien über 4 GB Größe.

5.5 Hilfe, es bootet nicht!

Grundsätzlich verschieden, wenn auch nicht unbedingt voneinander unabhängig, kann das LINBO-System selbst, oder das von ihm gestartete Betriebssystem der Auslöser für Startprobleme sein:

5.5.1 LINBO startet nicht / bleibt stehen

Wenn per PXE gebootet wird, und statt des Bootloaders eine Fehlermeldung ähnlich „File not found.“ erscheint, dann ist die Konfiguration des PXE-Bootladers (pxelinux.0) nicht korrekt, oder es kann vom Rechner, z.B. wegen

Netzwerk- oder Firmware-Problemen, nicht per TFTP auf die benötigten Startdateien des Servers zugegriffen werden. Einige TFTP-Server verweigern aufgrund von Sicherheitseinstellungen den Zugriff auf Dateien, die per Symlink („Verknüpfung“) in das Bootverzeichnis gelegt wurden, wenn sie sich tatsächlich außerhalb dieses Verzeichnisses befinden.

Dies kann ein Sicherheitsfeature des TFTP-Servers sein, um zu verhindern, dass das komplette Dateisystem durch einen falschen Symlink auf ein in der Hierarchie sehr weit vorne liegendes Verzeichnis (z.B. /) nach außen exportiert wird. Weiterhin kann in der DHCP-Konfigurationsdatei `dhcpd.conf` oder `pxelinux.cfg/*` ein falscher Dateiname oder Dateipfad für die beiden wesentlichen Dateien `linux/linx64` (Kernel) und `minirt.gz` (Initiale Ramdisk) angegeben sein, dann werden diese vom Bootloader selbst nicht gefunden. Linux-Dateisysteme sind grundsätzlich immer Groß-/Kleinschreibungs-sensitiv, auch dies muss beachtet werden.

Wird hingegen zunächst noch der LINBO-Kernel geladen, aber danach bleibt der Bildschirm schwarz oder das System bleibt mit einer „Kernel Panic“ stehen, dann ist höchstwahrscheinlich während der Hardware-Initialisierung ein Fehler aufgetreten. Der von LINBO verwendete Linux-Kernel enthält alle zum Start und zum Ansprechen der Hardware benötigten Module („Treiber“ im Windows-Jargon), diese werden nacheinander ausprobiert und versetzen die angesprochenen Hardwarekomponenten in einen funktionsfähigen Zustand. Leider sind bei den vielen verschiedenen Mainboards, CPUs und Peripherie-Chipsätzen hin und wieder Implementationsfehler vorhanden (d.h. Teile des Chipsatzes funktionieren gar nicht, oder werden vom Rechner-BIOS schon von vornherein falsch angesteuert), was auf Software-Seite durch Umgehung der nicht korrekt funktionierenden Komponenten „ausgetrickst“ werden kann. Unter Windows erledigen das sogenannte „Motherboard-Treiber“, die für jeden Rechner speziell vom Hersteller angeboten werden. Unter Linux sind hingegen die sogenannten Kernel-Bootoptionen zuständig, bestimmte Hardwarekomponenten einfach nicht zu nutzen bzw. zu ignorieren. Dabei werden diese nicht explizit dauerhaft „abgeschaltet“, sondern lediglich während der Laufzeit von Linux nicht weiter benutzt, und sind nach einem Reset des Rechners wieder so verfügbar, wie sie es vor dem Start von Linux auch waren. Auch die von LINBO gestarteten Betriebssysteme müssen sich nicht an die vom LINBO-Kernel zuvor „deaktivierten“ Komponenten halten. Allerdings kann es z.B. für den Windows-Boot per `kexec` eine Rolle spielen, ob sich bestimmte Komponenten beim Start in einem Grundzustand befinden, oder bereits einmal im „protected mode“ benutzt wurden. S.a. Rezept 57. Die Optionen in nachfolgender Tabelle können bei Bootproblemen helfen, und sind entweder direkt in der Konfigurationsdatei des OS-Bootloaders anzugeben oder können, wenn interaktives Starten von LINBO erlaubt ist, nachträglich im Bootscreen als Parameter nach `linbo` bzw. `linbo64` angegeben werden, z.B. für Tests. Es ist hingegen nicht möglich, einen fest in der Bootkonfiguration eingetragenen Parameter durch interaktive Eingabe wieder zu „löschen“, dies muss in der Konfigurationsdatei geschehen.

Beispiel für einen Eintrag im PXE-Bootloader, Datei
/tftpboot/pxelinux.cfg/default:

```
KERNEL linbo  
APPEND initrd=minirt.gz nfsdir=10.0.2.2:/var/linbo acpi=off intel_iommu=off
```

Hier wird angegeben, dass das „Advanced Configuration and Power Interface“ (ACPI) nicht verwendet wird, das bei modernen Computern auch die automatische Konfiguration von Erweiterungskarten anders als das BIOS dies vorgibt, erledigen kann, ebenso wird der IOMMU-Baustein für Controller auf Intel-Boards ignoriert, der bei einigen Computern nicht zuverlässig mit Linux funktioniert.

Bootparameter	Bedeutung
acpi=off	Abschalten des „Advanced Configuration and Power Interface“, Interrupts werden danach über andere Standard-Mechanismen zugewiesen. Dies kann helfen, wenn sich bestimmte Hardwarekomponenten „aufhängen“, oder das System ohne erkennbaren Grund stehenbleibt. Allerdings funktionieren bei einigen Rechnern USB-Geräte oder Netzwerkkarten nicht, wenn sie nicht ausschließlich per ACPI initialisiert worden sind.
acpi=noirq	Schaltet nur die Interrupt-Zuweisung durch ACPI ab. Powermanagement per ACPI bleibt aber eingeschaltet.
noapic	Schaltet den „Advanced Interrupt Controller“ auf dem Board ab, der auf neueren Boards das Interrupt-Handling übernimmt. Wenn der Rechner überhaupt nicht bis zur Hardware-Initialisierung kommt, ist der Grund manchmal ein falsch arbeitender APIC.
nolapic	Schaltet den lokalen „Advanced Interrupt Controller“ ab, der auf neueren CPUs das Interrupt-Handling übernimmt. Ähnlich noapic.
nolapic_timer	Schaltet nur den Zeitgeber des lokalen „Advanced Interrupt Controller“ ab. Entschärfte Form von nolapic.
nosmp	Benutzt nur die erste CPU bei Mehrprozessor-Systemen, und deaktiviert die Multiprozessor-spezifischen Verwaltungskomponenten. Für LINBO bedeutet dies keinen erheblichen Geschwindigkeitsverlust, da das (De-)komprimieren der Daten auf Festplatte zwar theoretisch mit mehr CPUs schneller durchgeführt wird, in der Praxis aber die Latenz der Festplatte der entscheidende Faktor bei der Restaurierungszeit ist (s.a. Abschnitt 7 Seite 64), worauf die Rechengeschwindigkeit einer oder mehrerer CPUs kaum Einfluss hat. Die durch

Bootparameter	Bedeutung
	LINBO gestarteten Betriebssysteme können von sich aus SMP wieder aktivieren, auch ohne dass ein Reset erforderlich ist.
pnpbios=off	Verhindert die Initialisierung von Komponenten durch das, eigentlich nur noch für ältere ISA-Karten vorhandene), „Plug & Play BIOS“
pci=bios	Verwendet ausschließlich die durch das Rechner-BIOS ein gestellten Interrupt-Tabellen für die Konfiguration von PCI-Karten.
debug	Zeigt ausführliche Boot- und Fehlermeldungen des Kernels an, und aktiviert das „Tracing“ mit Shell-Breakpoints. Nur zu Debugging-Zwecken für Entwickler sinnvoll.

5.5.2 Das von LINBO gestartete Betriebssystem startet nicht / bleibt stehen

LINBO versucht beim Klick auf einen der „Start“-Knöpfe, ggf. nach Synchronisation, bei Angabe von „Method = kexec“ in `start.conf`, das gewählte Betriebssystem direkt und ohne den sonst üblichen Reset des Rechners zu starten. Bei Linux ist dies über den `kexec()` System Call direkt möglich, bei Windows wird zusätzlich ein weiterer Bootloader, `grub.exe` zwischengeschaltet, der wiederum über spezielle Bootoptionen so konfiguriert wird, dass er den Windows-Bootloader als sog. „Chainloader“ startet.

Windows ist sehr empfindlich, was den Zustand der Hardware angeht, wenn der Windows-Kernel startet. Unter Linux werden die Hardwarekomponenten üblicherweise so konfiguriert, dass sie die für den Betriebszustand notwendigen Zustand einnehmen, auch wenn sie bereits zuvor ohne Unterbrechung in Betrieb waren.

Symptom: Beim Start von Windows erscheint noch eine Meldung der Art „Launching GRUB ...“ oder „Starting cmain...“, danach „hängt“ der Grub-Bootloader.

Erklärung: `grub.exe` findet die Festplatte nicht, da der Festplattencontroller sich in einem Zustand befindet, in dem er im sog. „real mode“ der CPU nicht mehr ansprechbar ist.

Mögliche Lösung: Einen alternativen Betriebsmodus des Festplattencontrollers im BIOS einstellen, sofern vorhanden, oder bestimmte Modulparameter für den Linux-spezifischen Controller-Treiber angeben. Dies ist eine komplizierte, und nicht immer erfolgreiche Lösung. I.d.R. muss bei diesem Problem aber auf eine andere Bootmethode zurückgegriffen werden, in der der Controller einen Zwangsreset durchführt, so wie dies bei einem

„Kaltstart“ des Rechners der Fall ist, d.h. es ist ein Reset über die Bootmethode „reboot“ erforderlich, bei der der Rechner schlichtweg neu gestartet wird, und dann statt LINBO das Betriebssystem direkt bootet.

Mitunter bringen auch die bereits im vorigen Abschnitt genannten Bootoptionen für LINBO Abhilfe, da sie die PCI-Konfiguration des problematischen Controllers verändern, und dadurch den „Soft-Reboot“ möglicherweise unterstützen. Dies können die Bootoptionen `acpi=noirq`, `nosmp`, `nolapic` oder `pci=bios` sein. Spezielle Bootoptionen wie `sata_nv.swncq=0`, die sich nur auf einen speziellen (in diesem Fall SATA-) Controller beziehen, können ggf. auch den Controller in den gewünschten Zustand bringen, und somit das Booten über `grub.exe` per `kexec` ermöglichen. Diese Bootparameter sind für den Fall Linux nach erfolgreichen Tests unter „Append = “ in `start.conf`, oder bei Windows in der `pxelinux`-Konfiguration von LINBO selbst zu setzen.

Symptom: Linux startet mit der `kexec`-Methode, aber der Bildschirm bleibt dunkel, bis die Grafikoberfläche erscheint.

Erklärung: LINBO aktiviert den beschleunigten Framebuffer für die Grafikkarte. Dieser kann ohne Hardreset nicht wieder zurück in den im „real mode“ üblichen VGA-Textmodus zurückgesetzt werden. Daher ist die Nutzung der normalen Textkonsole, oder auch Umstellen auf einen Framebuffer- oder VESA-Modus anderer Auflösung oder Farbtiefe nicht möglich. Erst im beschleunigten Grafikmodus, der beim Start des Xorg-Servers eingeschaltet wird, ist wieder ein Bild sichtbar.

Mögliche Lösung: Das zu bootende System so umstellen, dass je nach Fehlerart entweder kein Bootsplash-Bild, oder ein bestimmter Framebuffer-Modus verwendet wird („Append = `vga=791`“ Einstellung in `start.conf`). Alternativ muss auf die Bootmethode `reboot` zurückgegriffen werden, wenn die Grafikkarte nach `kexec` nicht wieder in einen funktionsfähigen Zustand kommt.

Symptom: Windows oder Linux starten zwar, aber irgendetwas ist falsch (z.B. die Maus bewegt sich nicht, oder das Netzwerk ist nicht erreichbar). Direkt ohne LINBO von Platte gestartet, funktioniert jedoch alles.

Erklärung: Das Abschalten oder Umkonfigurieren von Hardwarekomponenten durch den LINBO-Start ist nach dem „Soft-Reset“ bei `kexec` immer noch aktiv, und das gestartete Betriebssystem stellt nicht selbstständig den gewünschten Zustand der Hardwarekomponente wieder her, d.h. Interrupts laufen ins Leere oder IO-Adressen sind nicht erreichbar.

Mögliche Lösung: Für Linux als gestartetes Betriebssystem lässt sich durch explizite Angabe von z.B. `acpi=force` in der „Append =“ Zeile erreichen, dass bestimmte Eigenschaften re-aktiviert werden. Für Windows ist dies allerdings nicht ohne weiteres möglich, hier müssten für LINBO wiederum Boot-Optionen gefunden werden, die das Booten ermöglichen, aber keine Auswirkungen auf das gestartete Betriebssystem haben. In einem Beispielszenario auf einem ASUS-Notebook bewirkten sowohl die APPEND-

Optionen `acpi=noirq` als auch `nosmp` in der `pxlinux.cfg/default` Pxe-Bootkonfiguration das einwandfreie Booten von Windows mit der `kexec`-Methode, allerdings funktionierte nur mit `nosmp` der integrierte USB-Controller. Mit `acpi=noirq` wurden hingegen USB-Maus und USB-Tastatur nicht von LINBO erkannt. Mit Zurückgreifen auf die Bootmethode `reboot` traten auch in diesen Fällen keine Probleme mehr auf.

5.6 Wie erzeuge ich ein virtuelles System mit und für virtualbox, zur Benutzung durch LINBO?

Die Open Source Virtualisierungssoftware **VirtualBox** (ehemals innotek, heute vertrieben durch Firma Oracle) verwendet zur Speicherung von OS-Daten zwei Dateiformate:

1. `OSNAME.vbox`: Hier werden im XML-Format alle im Virtualisierer eingestellten Parameter abgelegt, dies betrifft v.a. die simulierte Hardware des Rechners.
2. `OSNAME.vdi` (+ ggf. weitere `.vdi`-Dateien): Dieses Image-Format ist ein sog. „Sparse File“, es enthält die Festplatten und andere Datenträger der virtuellen Maschine, jedoch werden noch nicht benutzte Sektoren, die i.d.R. aus Bytes mit dem Wert 0 bestehen, weggelassen. Die Datei hat daher „Löcher“ und nimmt nur einen Bruchteil ihrer im Dateilisting angezeigten Größe tatsächlich an physikalischem Platz ein. Solche Dateien werden von `rsync` unter LINBO speziell behandelt, damit nicht beim Kopieren die null-Bytes, also die „Löcher“ in der Datei, mit Nullen aufgefüllt und die Datei auf ihre Maximalgröße expandiert wird.

Beide Formate befinden sich in einem Verzeichnis `OSNAME`, das Virtualbox standardmäßig im Ordner „Virtualbox VMs“ im Heimverzeichnis des Anwenders ablegt.

Grundsätzlich lässt sich die Virtuelle Maschine auch ohne LINBO auf einem beliebigen System mit installierter Virtualbox-Software herstellen.

Unter LINBO sind sowohl die 32bit-Version von Virtualbox, als auch die 64bit-Version (in einer `chroot`-Umgebung) installiert. Im Gegensatz zu `kvm` muss bei Virtualbox das Gast-Betriebssystem die gleiche Bit-Architektur besitzen wie Virtualbox selbst, d.h. für eine Installation von Windows oder Linux in 64bit muss die 64bit-Variante von Virtualbox gestartet werden.

Um Virtualbox in 64bit unter LINBO zu starten, muss

1. LINBO auf einer 64bit-fähigen CPU gestartet werden,
2. der 64bit-Kernel aktiviert sein (`linbo64`),
3. `virtualbox` aus einer 64bit `chroot`-Umgebung gestartet werden (Terminal):
`linbo_cmd shell64`
`virtualbox`

Die virtuelle Maschine sollte die hardwarespezifischen Merkmale, die durch

Paravirtualisierung direkt vom Hostsystem an das Gastsystem durchgereicht werden, durch entsprechende Treiber bzw. Kernel-Konfiguration abbilden, damit nicht beim Start auf anderer Hardware eine erneute Treiber-Installation (speziell bei Windows) angestoßen wird. Außerdem sollten die Gast-Erweiterungen für beschleunigte Grafik und Durchreichen von bestimmten Hardwarekomponenten für Virtualbox auf dem Gastsystem installiert werden, die von der verwendeten Virtualbox-Version abhängig sind.

Wenn LINBO eine virtuelle Maschine startet, wird der virtualisierte RAM-Speicher der Größe des Hostsystems angepasst, einige Einstellungen wie Fullscreen-Modus und Durchreichen von USB-Geräten per Interaktion mit dem Menü eingeschaltet, und die virtuelle Festplatte wird, außer im „Modify“-Modus, durch das Anlegen eines temporären Snapshots auf der schreibbaren Cache-Partition, in den „Read-Only“-Modus versetzt, so dass nach einem Neustart der VM der Ursprungszustand wieder hergestellt wird.

Für den Start virtueller Maschinen ist also keine Synchronisation notwendig, außer, wenn sich auf dem Server der Inhalt des VM-Verzeichnisses geändert hat, was durch einen Vergleich der zugehörigen *OSNAME.info*-Dateien festgestellt wird.

6 LINBO Backend – API

Das LINBO Backend wird zum größten Teil durch das Worker-Skript `/usr/bin/linbo_cmd` realisiert, das in der initialen Ramdisk und vom GUI aufgerufen wird. Die nachfolgende Tabelle listet die Befehle, Optionen und ihre Wirkung auf. Anders als in LINBO V2 erhält `linbo_cmd` seine Konfigurationsdaten durch direktes Parsen der Datei `/start.conf`. Die Kommandos können auch im Terminal-Fenster durch den unprivilegierten User aufgerufen werden, `linbo_cmd` führt dabei automatisch alle privilegierten Aktionen mit `root`-ID aus.

Kommando	Beschreibung
<code>linbo_cmd server</code>	Ausgabe der IP-Adresse des rsync-Servers
<code>linbo_cmd ip</code>	Ausgabe der aktuellen IP-Adresse des Client
<code>linbo_cmd wlan</code>	Auslesen der WLAN-Konfiguration aus <code>start.conf</code> und Konvertieren in das <code>/etc/network/interfaces</code> -Format (wird derzeit nicht verwendet)
<code>linbo_cmd hostname</code>	Ausgabe des per DHCP oder DNS gesetzten Hostnamens des Client
<code>linbo_cmd cpu</code>	Ausgabe der CPU-Informationen des Client
<code>linbo_cmd memory</code>	Ausgabe des Größe des Hauptspeichers in MB
<code>linbo_cmd mac</code>	Ausgabe der aktiven MAC-Adresse
<code>linbo_cmd size partition</code>	Ausgabe der Größe der angegebenen Partiton in kB
<code>linbo_cmd authenticate server user password</code>	Anmeldeversuch auf dem schreibbaren rsync-Share zur Authentifizierung, liefert „true“ bei Erfolg, „false“ bei Fehlschlag
<code>linbo_cmd create OSName</code>	Erzeugt die in <code>start.conf</code> angegebenen komprimierten Images für das angegebene Betriebssystem neu
<code>linbo_cmd start OSName VM</code>	Startet das angegebene native Betriebssystem bzw. die VM ohne Synchronisation
<code>linbo_cmd modify_vm</code>	Startet die angegebene VM im schreibbaren Modus, ohne Snapshot-

Kommando	Beschreibung
	Overlay. Alle Modifikationen wirken sich direkt auf die VDI-Dateien aus
<code>linbo_cmd checkpartitions</code>	Überprüft, ob alle Partitionen gemäß <code>start.conf</code> entsprechend eingerichtet sind, liefert „true“ wenn Ja, „false“ wenn Nein.
<code>linbo_cmd initpartitions</code>	Stellt die Partitionierung gemäß <code>start.conf</code> ohne Rückfrage her, vorhandene Daten werden überschrieben!
<code>linbo_cmd initpartitions_interactive</code>	Stellt die Partitionierung gemäß <code>start.conf</code> her, nach vorheriger Abfrage als Dialog (Bestätigung erforderlich)
<code>linbo_cmd initcache</code>	Formatiert die Cache-Partition und markiert sie durch eine versteckte Datei „.linbo-cache“ zum späteren automatischen Finden in der initialen Ramdisk
<code>linbo_cmd mountcache [-r]</code>	Bindet die Cache-Partition (bei -r read-only, sonst schreibbar) ein
<code>linbo_cmd syncl OSName [full]</code>	Synchronisiert aus einem <i>lokal vorhandenen</i> Image auf der Cache-Partition auf die Zielpartition. Falls <code>full</code> angegeben ist, wird ein ggf. in <code>start.conf</code> angegebener Quicksync-Eintrag für die jeweilige Partition ignoriert
<code>linbo_cmd syncr OSName</code>	Prüft auf dem Server nach einer neuen Version eines Image und aktualisiert ggf. das Image auf der Cache-Partition
<code>linbo_cmd syncstart OSName [full]</code>	Kombination aus <code>syncr</code> , <code>syncl</code> und <code>start</code>
<code>linbo_cmd update_linbo</code>	Aktualisiert die zum lokalen Start von LINBO notwendigen Daten im Cache vom Server und installiert ggf. den MBR neu
<code>linbo_cmd update_images</code>	Aktualisiert <u>alle</u> Images aus <code>start.conf</code> auf der Cache-Partition vom Server

Kommando	Beschreibung
<code>linbo_cmd update_startconf</code>	Aktualisiere <code>/start.conf</code> vom rsync-Server
<code>linbo_cmd syncall</code>	Aktualisiert alle Images aus <code>start.conf</code> auf der Cache-Partition vom Server und führt anschließend eine lokale Synchronisation der Dateisysteme aus den Images durch (per <code>sync</code>)
<code>linbo_cmd upload user password file</code>	Lädt die angegebene Datei auf den rsync-Server hoch (mit Authentifizierung am linbo-upload Repository)
<code>linbo_cmd upload_images user password osname</code>	Lädt alle Images, die laut <code>start.conf</code> zu einem Betriebssystem gehören, auf den rsync-Server hoch (mit Authentifizierung am linbo-upload Repository)
<code>linbo_cmd version</code>	Gibt die Versionen von Kernel, initialer Ramdisk und Builddatum des komprimierten LINBO-Dateisystems aus
<code>linbo_cmd resize partition [new_size]</code>	Ändert, wenn möglich, die Partitionsgröße. Bei Angabe von <code>new_size</code> auf die angegebene Größe (in kB), ansonsten gemäß der Angabe für die Partition in <code>start.conf</code>
<code>linbo_cmd shell64</code>	Startet eine interaktive Shell in der 64bit chroot-Umgebung (funktioniert nur, wenn mit 64bit-Kernel linbo64 gebootet wurde)
<code>linbo_cmd patch_system partition patchfile</code>	Wendet die angegebene Registry-Patchdatei auf eine Partition an
<code>linbo_cmd patch_vm VMNAME</code>	Wendet die in <code>start.conf</code> vermerkte Registry-Patchdatei auf die Systempartition in der VDI-Datei der virtuellen Maschine an.
<code>linbo_cmd help</code>	Gibt eine Kurzhilfe zu <code>linbo_cmd</code> aus

7 rsync - physikalische Limits

Während bei der Übertragung von großen Datenmengen, z.B. bei der Neuinstallation, die Lese-/Schreibgeschwindigkeit der Festplatte die größte Rolle spielt, wird die Geschwindigkeit beim Synchronisieren („Reparieren“) des Dateisystems im wesentlichen durch das Vergleichen von bestimmten Metadaten, v.a. Zeitstempel und Dateigröße, zwischen Quell- und Zielfilesystem bestimmt. Die zum Dekomprimieren der Daten benötigte Rechenzeit spielt auf modernen Computern kaum eine Rolle, tatsächlich beschleunigt die cloop-Kompression den Vorgang, da weniger Daten physikalisch von der Festplatte gelesen werden müssen.

Da jede einzelne Datei auf der Zielpartition auf Veränderungen gegenüber dem Original untersucht werden muss, spielen Anzahl und Menge der Daten eine geringere Rolle als der physikalische Parameter *seek time*, d.h. die Zeit, die der Schreib/Lesekopf der Festplatte benötigt, um an die Stelle positioniert zu werden, an der sich die Metadaten der untersuchten Datei befinden.

Da Quell- und Zielfilesystem miteinander verglichen werden müssen, sind im günstigsten Fall (die beiden Dateien befinden sich auf einem wenig fragmentierten Dateisystem mit einer optimal geringen Anzahl zu durchsuchender Knoten des Dateisystembaums, keine Änderung der Daten) zwei *seek*-Operationen pro Zielfilesystem erforderlich. Erst, wenn tatsächlich ein Unterschied zwischen Quell- und Zielfilesystem festgestellt wird, wird ein Kopiervorgang mit anschließendem Transfer der erweiterten Dateiattribute durchgeführt.

$$T_{\min} \approx N_{\text{nodes}} \times 2 \times t_{\text{seek}} + C_{\text{nodes}} \div (S_{\text{read}} \times K_{\text{cloop}}) + C_{\text{nodes}} \div S_{\text{write}}$$

T_{\min} : (Minimale) Gesamtzeit für die Synchronisation

t_{seek} : Seek time/delay der Festplatte

N_{nodes} : Anzahl zu untersuchender Knoten (Dateien/Verzeichnisse)

C_{nodes} : Summe Dateigröße(n) aller geänderten Dateien

S_{read} : Lesegeschwindigkeit der Festplatte

S_{write} : Schreibgeschwindigkeit der Festplatte

K_{cloop} : cloop-Kompressionsfaktor (typisch c.a. 3:1 Kompression)

Beispielrechnung:

Eine bereits vorhandene Windows-Installation mit vielen zusätzlich installierten Programmen enthält 150.000 Knoten (=Dateien und Verzeichnisse) und soll per rsync aus dem eingebundenen cloop-Archiv

synchronisiert werden. Die Festplatte besitzt eine verhältnismäßig gute *seek-time* von nur 5ms (Millisekunden) per Seek. Ohne dass eine Datei geändert wurde, beträgt die Zeit, die für einen Synchronisationsdurchgang benötigt wird, rechnerisch:

$$\begin{aligned} T_{\min} &= 150.000 \times 2 \times 5\text{s}/1000 \\ &= 1500\text{s} \\ &= 25\text{min} \end{aligned}$$

Bei einer „Minimalinstallation“ von Windows sind zwischen 60.000 und 80.000 Knoten zu überprüfen, dementsprechend reduziert sich die Synchronisationszeit fast linear auf die Hälfte oder weniger.

Durch den Linux-typischen *Read-Ahead* und das *Caching* des Dateisystems können einige der Seek-Vorgänge eingespart werden, so dass sich die benötigte Zeit durchaus halbieren oder sogar dritteln kann, dies hängt jedoch stark von der Fragmentierung und der Art des Dateisystems ab. SSD-Festspeicher sind durch die fast komplett entfallende *seek time* beim dateiweisen Synchronisieren klar im Vorteil.

Beim neu Anlegen des Dateisystems und Kopieren von Dateien defragmentiert Linux das Zielsystem automatisch und schreibt sequentiell Dateien hintereinander, mit gelegentlicher Aktualisierung der Metadaten. Hierdurch entfallen viele Seeks, so dass sich die Gesamtzeit für die Kopie reduziert und im Wesentlichen noch durch die Schreib-/Lesegeschwindigkeit der Festplatte bestimmt wird.