

LINBO

Linux-basiertes Interaktives Netzwerk-Bootsystem

Klaus Knopper

1. August 2007

Inhaltsverzeichnis

1	Einführung	5
1.1	Funktionsweise / Technik	5
1.2	Testlauf in qemu	6
1.3	Namen und Beschreibungen	6
1.3.1	Cache-Partition	6
1.3.2	Multicast	6
1.3.3	PXE	7
1.3.4	cloop	7
1.3.5	rsync und das rsync-Batch-Format	7
2	Installation	8
2.1	Bootvorgang	8
2.2	LINBO- und Dateisystem-Konfiguration	8
3	Anwenderhandbuch	10
3.1	LINBO booten	10
3.2	Graphische LINBO Oberfläche	11
3.2.1	Betriebssysteme wiederherstellen und starten	11
3.2.2	Betriebssystem-Images verwalten	13
4	Administration	14
4.1	Installation und Konfiguration	14
4.1.1	start.conf - Partitionen und Images	14
4.1.2	PXE-Konfiguration (DHCP-Server)	15
4.1.3	RSYNC-Konfiguration (Server)	17
5	LINBO-Buildsystem	20
5.1	Systemvoraussetzungen	20
5.2	Verzeichnisse	20

5.3	Bauvorgang	21
6	Das LINBO-Kochbuch, „How do I ...?“	22
6.1	Der allererste Start: Wie richte ich ein Master-System für LINBO-Images ein?	22
6.2	Was muss in der start.conf stehen?	23
6.3	Wie groß soll die Cache-Partition sein, und wo genau soll sie auf der Festplatte liegen?	23
6.4	Wie setzen sich die Partitionsnamen unter Linux zusammen, was muss ich angeben?	23

Abbildungsverzeichnis

1	UML Aktivitätsdiagramm für LINBO	5
2	PXE-Bootlader in qemu	10
3	Start von LINBO über pxelinux	10
4	Startmenü von LINBO	11
5	Neu aufsetzen eines Betriebssystems mit LINBO	12
6	Aus LINBO gestartetes Mini-Linux	12
7	Neu aufsetzen - Dekompressionsvorgang	13
8	Beispiel für grub.exe.info	19
9	Kopieren der LINBO-Dateien ins rsync-Repository	19
10	Einloggen als Admin bei LINBO	24
11	Image von Partition erzeugen, Dateiname	25
12	Image von Partition erzeugen, Kompression	26
13	Image von Partition erzeugen, Upload	27
14	Neu aufsetzen (1)	28
15	Neu aufsetzen (2)	29
16	Sync+Start	30
17	OS-Boot	31
18	Partitionieren	32

1 Einführung

LINBO ist ein halb- bis vollautomatisch (je nach Konfiguration) arbeitender Bootmanager, der nicht nur in der Lage ist, verschiedene Betriebssysteme von Festplatte zu starten, sondern der auch Wartungs-, Update- und Reparaturfunktionen für Festplatteninstallationen übernimmt.

1.1 Funktionsweise / Technik

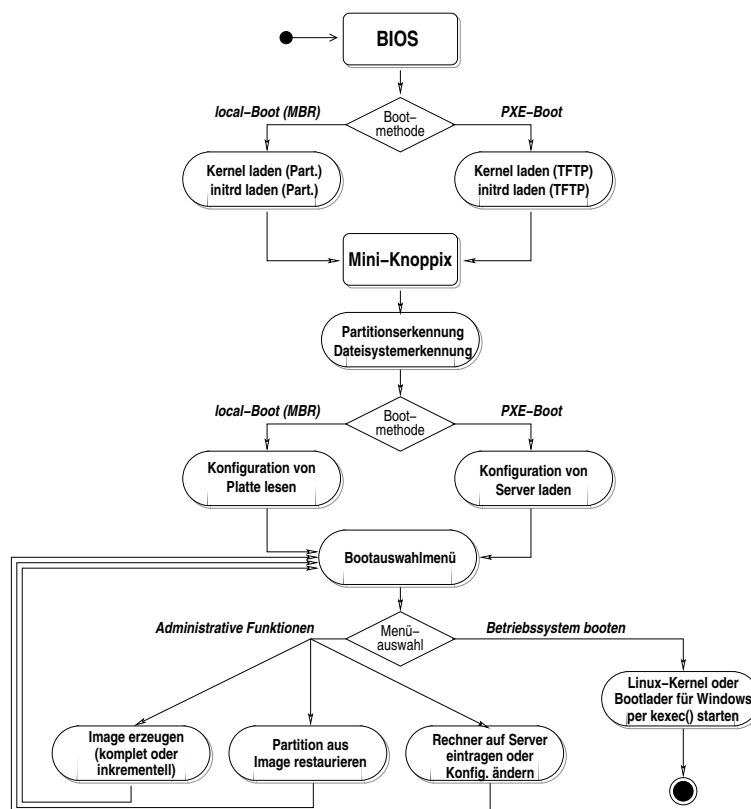


Abbildung 1: UML Aktivitätsdiagramm für LINBO

Vom Bootlader (lilo lokal von Platte, oder Netzwerk PXE/Bootp-Server) werden Kernel und initiales Ram-Dateisystem geladen und gestartet. Nach normalerweise recht kurzer Startzeit wird eine graphische Oberfläche, **linbo_gui** mit Auswahlmöglichkeit präsentiert, während parallel dazu die Hardwareerkennung von Netzwerkkarten und Festplatten läuft. Nach Auswahl eines Buttons werden verschiedene Aktionen über das Worker-Backend-Skript **linbo_cmd** abgewickelt.

1.2 Testlauf in qemu

(Installation in einer richtigen DHCP-Server-Umgebung: Siehe Abschnitt 4.1.2.)

Das zentrale **Makefile** im LINBO-Entwicklungsverzeichnis bietet drei Testszenarien mit Hilfe von **qemu** als virtuelle Machine(n):

make test	Direktes Booten von LINBO-Kern und Start des internen TFTP-Servers (Directory „Images“)
make hctest	Booten von simulierter Festplatte Images/hda.img
make pxetest	Booten per PXE von einem durch qemu simulierten PXE-Server

Achtung: Für die zuletzt genannte Option ist die Installation von qemu ab Version 0.9.0+cvs erforderlich, da frühere Versionen den „bootp“-Parameter noch nicht kannten! Im Buildsystem befindet sich ein aktueller Snapshot von qemu, der mit **make qemu** als Debian-Paket gebaut wird (gcc-3.4 erforderlich).

1.3 Namen und Beschreibungen

1.3.1 Cache-Partition

Auf den mit LINBO verwalteten Rechnern wird eine Cache-Partition verwendet, um LINBO selbst und die verwalteten Betriebssysteme lokal vorzuhalten, und notfalls auch ohne Netzwerk starten zu können.

1.3.2 Multicast

Um den Cache mit den großen Image-Dateien (komprimiert ca. 500MB-2GB pro Betriebssystem je nach Ausstattung) effizient zu füllen, kann optional Multicast verwendet werden. Hierzu muss auf dem LINBO-Server **udpcast** installiert sein, welches nach einer Mindestanzahl anfordernder Clients und einstellbarer Wartezeit das Senden der Images an mehrere Rechner gleichzeitig unterstützt. Hierdurch werden die Daten nur einmal physikalisch übertragen, wenn mehrere Clients gleichzeitig den Cache mit Daten füllen, wodurch der Zeitaufwand beim erstmaligen Installieren oder Update von Clients drastisch reduziert wird. Beim **Sync+Start** schaut hingegen jeder Client auf dem Server nach einem Update des gewählten Images, und überträgt die Änderungen zur älteren Version per RSYNC. Sind keine Änderungen vorhanden, so wird die Version aus dem Cache weiterverwendet.

1.3.3 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 erforderlich, oder eine entsprechende Bootdiskette mit Treiber von <http://www.rom-o-matic.net>.

1.3.4 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 Datei-Form enthält. In LINBO haben diese Dateien die Endung **.cloop**. Im Gegensatz zu den bekannten **zip** oder **tar.gz**-Archiven verhält sich ein über *cloop* eingebundenes Archiv wie eine Festplattenpartition mit wahlfreiem Zugriff, die enthaltenen Daten und Teile davon werden „on demand“ dekomprimiert. In diesem Dateiformat ist es möglich, komplette Festplattenpartitionen mit allen Zusatzdaten wie *Boot-Record* und „versteckten“ Informationen leicht zugänglich zu halten. Auch das Herauskopieren einzelner Dateien ist dadurch möglich. In LINBO werden alle Basis-Images (direkte Partitionsabzüge) in diesem Format unverändert gespeichert, was auf der Cache-Partition Platz spart und den Lesevorgang dadurch, dass weniger physikalische Lesezugriffe erfolgen, 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, und bei Zufallsdaten, verschlüsselten Dateien oder bereits komprimierten Bildern ca. 1:1 bis 0,9:1.

1.3.5 rsync und das rsync-Batch-Format

rsync ist ein Synchronisierungs-Programm, das eine Kopie so ausführt, dass nur die *Änderungen* zwischen Quelle und Ziel übertragen werden. Statt von einem Quellverzeichnis zu einem Zielverzeichnis zu kopieren, unterstützen neuere Versionen von *rsync* das sog. „Batch“-Format, was besser mit „Binärdifferenz-Archiv“ übersetzt werden kann. In diesem Dateiformat werden die Differenzen zum Originalverzeichnis inklusive zu löschender Dateien gespeichert, optional ebenfalls wie bei *cloop* auch komprimiert, so dass es sich hervorragend für inkrementelle Archive eignet. LINBO legt inkrementelle Partitions-Images in diesem Format ab in Dateien mit der Endung **.rsync**. Diese können leider, im Gegensatz zum mountbaren *cloop*-Format, nur von *rsync* verarbeitet werden.

2 Installation

LINBO wird üblicherweise per PXE gebootet, und kann sich selbst auf die Cache-Partition ([1.3.1](#)) kopieren, und anschließend auch standalone von dort booten (d.h. ohne Netz).

2.1 Bootvorgang

LINBO kann wie ein normaler Linux-Kernel gebootet werden, unabhängig ob von lokaler Festplattenpartition oder einem PXE/BOOTP-fähigen DHCP-Server.

Aus technischen Gründen¹ ist LINBO aufgesplittet in einen Kernel-Teil **linbo** (ca. 2-3MB komprimiert), und einen Dateisystem-Teil **linbofs.gz** (ca. 8MB komprimiert), wobei der Kernelteil auch ein kleines Dateisystem mit **busybox** als Minimalshell, und der Dateisystem-Teil die größeren Programme und Tools wie **linbo_gui**, Systembibliotheken, und die Initialkonfiguration **start.conf** enthält.

linbo und **linbofs.gz** werden für den Netzwerk-Boot üblicherweise auf dem DHCP+TFTP-Server installiert, siehe auch Abschnitt [4.1.2](#).

2.2 LINBO- und Dateisystem-Konfiguration

LINBO erhält seine Boot- und Konfigurationsdaten über folgende Methoden:

1. Bootparameter, die sich per DHCP/pxelinux setzen lassen, diese sind:

ip=ip-adresse FESTE IP-Adresse (wenn gewünscht) für diesen Client

server=ip-adresse IP-Adresse des Servers, der die Images vorhält

cache=/dev/Partitionsname (s.a. Abschnit [6.4](#)) Partition, die Betriebssystem-Images vorhält

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.

¹Es hat sich in Tests gezeigt, dass einige Netzwerkkarten keine Einzeldateien größer 8MB per TFTP beziehen können, außerdem ist der absolute Adressraum für den Kernel auf wenige MB begrenzt

2. Aus der Datei **start.conf-ip-adresse**, die auf dem Server per rsync-Download angeboten wird. *ip-adresse* ist die für diesen Client per Bootkommandozeile oder per DHCP festgelegte IP-Adresse. Hiermit kann für jeden Rechner eine spezielle Konfiguration vereinbart werden. Um Gruppen von Rechnern mit dergleichen Konfiguration zu definieren, genügt es, einen Symlink (Beispiel:

```
ln -s start.conf-Klasse1A start.conf-192.168.0.2
```

) auf eine gemeinsame Konfigurationsdatei (**start.conf-Klasse1A** in diesem Beispiel -Groß- und Kleinschreibung werden beachtet!) anzulegen.
3. Aus einer Datei **start.conf**, die auf dem Server per rsync-Download angeboten wird. Dies ist quasi die „Default“-Einstellung, wenn weder per Bootkommandozeile, noch per Rechnerspezifischer **start.conf**-Datei Einstellungen vorgenommen werden.
4. Aus einer Datei **start.conf** auf der Cache-Partition des Client-Rechners.
5. Aus einer im LINBO-Dateisystem **linbofs.gz** integrierten **start.conf**-Datei. Dies ist der Fallback, wenn kein Rsync-Server vorhanden und noch keine Cache-Partition eingerichtet ist.

Der Aufbau der **start.conf**-Datei ist in Abschnitt [6.2](#) genau beschrieben.

3 Anwenderhandbuch

3.1 LINBO booten

LINBO kann sowohl übers Netz per PXE (1.3.3) als auch von einer bereits mit LINBO installierten Festplatte gestartet werden. Die Installation eines Bootservers für LINBO ist unter 4.1.2 beschrieben, die Installation des LINBO-Bootladers auf Festplatte unter 4.1.3.

```
Boot from (N)etwork or (Q)uit? N
Relocating _text from: [00000000,00000000] to [07000000,07000000]
Boot from (N)etwork or (Q)uit? N

Probing pci nic...
[rtl8029]
NE2000 base 0xc100, addr 52:54:00:12:34:56
Searching for server (DHCP)....
Me: 10.0.2.15, DHCP: 10.0.2.2, TFTP: 10.0.2.2, Gateway 10.0.2.2
Loading 10.0.2.2:/pxelinux.0 ..(PXE).....done

PXELINUX 3.31 Debian-2007-03-09 Copyright (C) 1994-2005 H. Peter Anvin
UNDI data segment at: 00000000
UNDI data segment size: 1000
UNDI code segment at: 00000000
UNDI code segment size: 0A00
PXE entry point found (we hope) at 9F00:0600
My IP address seems to be 0A00020F 10.0.2.15
ip-10.0.2.15:10.0.2.2:10.0.2.2:255.255.255.0
TFTP prefix: /
Trying to load: pxelinux.cfg/01-52-54-00-12-34-56
Trying to load: pxelinux.cfg/0A00020F
Trying to load: pxelinux.cfg/0A00020F
Trying to load: pxelinux.cfg/0A00020F
Trying to load: pxelinux.cfg/0A00020F
```

Abbildung 2: PXE-Bootlader in qemu

LINBO besteht aus einem Kernel- und einem Dateisystem-Teil, die separat geladen und anschließend automatisch im Hauptspeicher zusammengesetzt werden. Nach einer minimalen Hardwareerkennung (i.e. Grafikkarte) durch den Kernel, wird die graphische Oberfläche von LINBO, **linbo_gui**, gestartet.



Abbildung 3: Start von LINBO über pxelinux

Hinweis: Da parallel zum Start der Oberfläche eine weitere Hardwareerkennung

stattfindet (Netzwerk, Festplattencontroller und -partitionen), stehen einige LINBO-Funktionen erst nach einigen Sekunden zur Verfügung. Normalerweise ist das vom GUI aufgerufene linbo_cmd Worker-Backend aber so intelligent, dass es bei noch nicht erkannten Festplattenpartitionen einige Zeit wartet, bis diese verfügbar sind.

3.2 Graphische LINBO Oberfläche

3.2.1 Betriebssysteme wiederherstellen und starten

Abbildung 4 zeigt das Startmenü von LINBO. Hier sind alle Betriebssysteme, die für LINBO vorbereitet und auf Festplatte installiert wurden, aufgeführt.

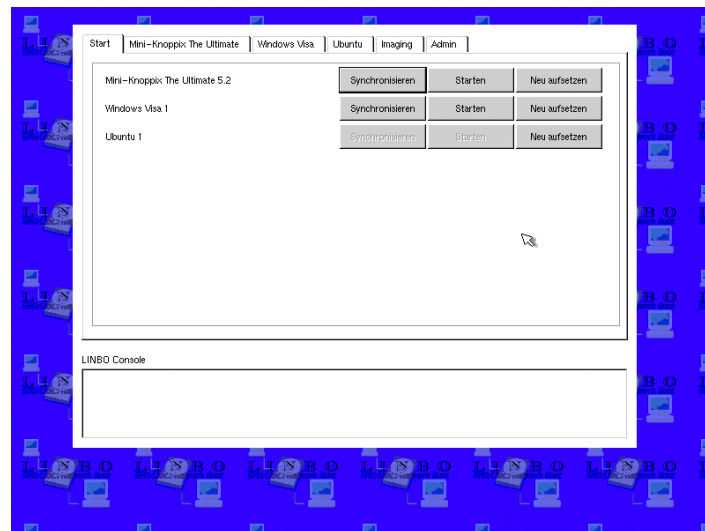


Abbildung 4: Startmenü von LINBO

Mit Klick auf **Synchronisieren** hinter dem Namen des Betriebssystems, wird das auf einer Partition befindliche System mit Hilfe eines auf der Cache-Partition (1.3.1) befindlichen Archivs überschrieben bzw. in den Ursprungszustand versetzt. *Achtung: Hierbei gehen alle Änderungen, die in der letzten Session mit diesem Betriebssystem erstellt wurden, verloren.*

Neu aufsetzen (Abbildung 5) lädt eine ggf. neuere Version des jeweiligen Betriebssystems vom Server per TFTP/Multicast auf die Cache-Partition herunter, und installiert diese anschließend wie bei **Synchronisieren**.

Start bootet das angegebene Betriebssystem so, wie es sich derzeit auf der Festplatte befindet. Der Rechner startet hierbei nicht neu, sondern LINBO führt einen

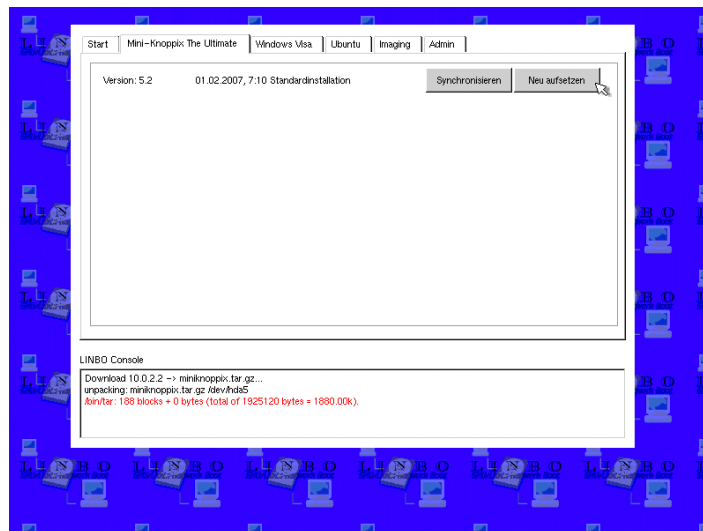


Abbildung 5: Neu aufsetzen eines Betriebssystems mit LINBO

„Soft-Reboot“ durch, was den Startvorgang stark beschleunigt. Treten beim Starten Fehler auf, oder befindet sich das installierte Betriebssystem nicht mehr in einem benutzbaren Zustand, so sollte nach dem nächsten Reboot mit Hilfe von **Synchronisieren** oder **Neu aufsetzen** wieder der zuletzt gespeicherte, arbeitsfähige Zustand restauriert werden, bevor ein neuer **Start** versucht wird.

Abbildung 6 zeigt ein Mini-Linux, das durch LINBO gestartet wurde.

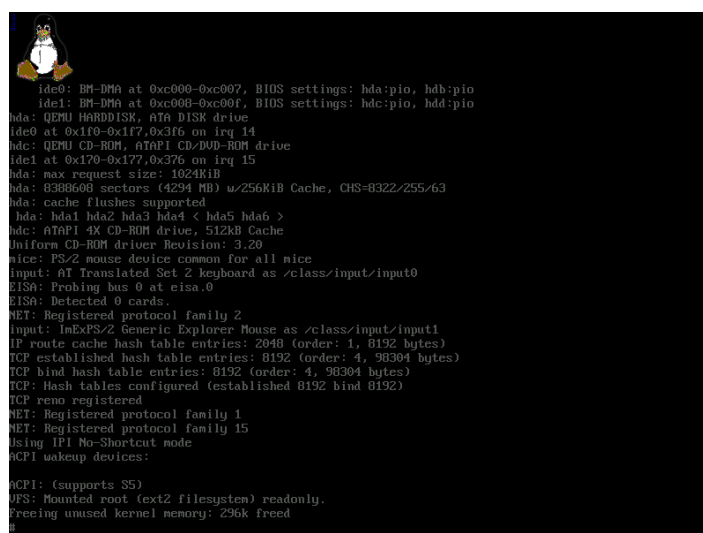


Abbildung 6: Aus LINBO gestartetes Mini-Linux

3.2.2 Betriebssystem-Images verwalten

Die „Reiter“ hinter dem Startmenü sind für die Verwaltung von Images (Archiven) der installierten oder zu installierenden Betriebssysteme zuständig. Hier können verschiedene Versionen eingespielt werden, die inkrementell auf einem Basis-Image aufbauen.

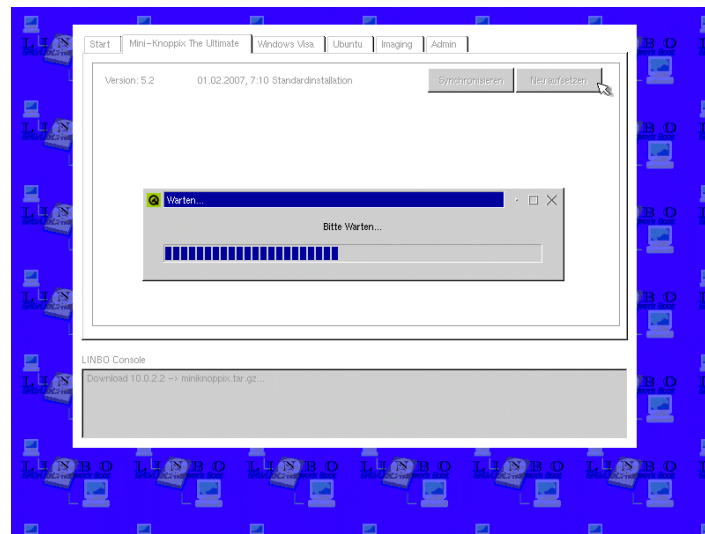


Abbildung 7: Neu aufsetzen - Dekompressionsvorgang

Gegenüber „Synchronisieren“ und „Neu aufsetzen“ aus dem Startmenü erlauben die gleichnamigen Buttons in den Betriebssystem-Reitern also eine genauere Angabe der jeweiligen Version, während im Startmenü immer nur die neueste Version restauriert wird.

4 Administration

4.1 Installation und Konfiguration

4.1.1 start.conf - Partitionen und Images

Die im **linbofs.gz** für jeden Client-Rechner befindliche Datei **start.conf** ist im Stil der bekannten KDE-Desktop-Iconbeschreibungen verfasst. Kommentare werden durch **#** eingeleitet, dürfen am Anfang einer Zeile oder mitten im Text auftauchen, und werden inklusive bis zum Zeilenende folgendem ext von **linbo_gui** ignoriert.

Eine Beispieldatei, die sich im Buildsystem in **Binaries/linbo_gui/start.conf** befindet, ist hier angegeben.

```
[LINBO]                                # LINBO global config
Cache = /dev/hda2                      # Cache Partition with local Images
Server = 10.0.2.2                      # First TFTP Server with remote Images

[Partition]                            # Start of a Partition config section
Dev = /dev/hda1                        # Device name of partition (hda1 = first partition on first IDE disk)
Size = 4200997                         # Partition size in kB
Id = 7                                # Partition type (83 = Linux, 82 = swap, c = FAT32, 7 = NTFS, ...)
FSType = ntfs                          # File system on Partition
Bootable = yes                        # Mark this partition as bootable

[Partition]                            # Device name of partition
Dev = /dev/hda2                        # Partition size in kB
Size = 4096575                         # Partition type (83 = Linux, 82 = swap, c = FAT32, ...)
Id = 83                                # File system on Partition
FSType = reiserfs                      # Mark this partition as non-bootable (or Linux)
Bootable = no

[Partition]                            # Device name of partition
Dev = /dev/hda3                        # Partition size in kB
Size = 1052257                         # Partition type (83 = Linux, 82 = swap, c = FAT32, ...)
Id = 82                                # File system on Partition
FSType = swap

[Partition]                            # Device name of partition
Dev = /dev/hda4                        # Partition size in kB (empty if "remaining space")
Size =                                 # Partition type (5 = Extended)
Id = 5                                 # File system on Partition (none for extended partition)
FSType =                               # Mark this partition as non-bootable (or Linux)
Bootable = no

[Partition]                            # Device name of partition
Dev = /dev/hda5                        # Partition size in kB (empty if "remaining space")
Size =                                 # Partition type (83 = Linux, 82 = swap, c = FAT32, ...)
Id = 83                                # File system on Partition
FSType = ext2                          # Mark this partition as non-bootable (or Linux)
Bootable = no

[OS]
Name = $\mu$-Knoppix The Ultimate      # Name of OS
```

```

Version = 5.2          # Version/Date of OS (optional)
Description = 01.02.2007, 7:10 Standardinstallation # Descriptive Text
Image =                # Filename of rsync batch, empty for none
BaseImage = microknoppix.cloop # Filename of base partition image
Boot = /dev/hda5       # Partition containing Kernel & Initrd
Root = /dev/hda5       # root=/dev/partition Parameter (Root FS)
Kernel = vmlinuz       # Relative filename of Kernel or Boot image
Initrd =               # Relative filename of Initrd
Append =               # Kernel cmdline, root= will be added (optional)
StartEnabled = yes     # Enable "Start" Button
SyncEnabled = yes     # Enable "Synchronize" Button
RemoteSyncEnabled = yes # Enable "Synchronize from Server" Button
Autostart = yes        # Boot this OS by default

[OS]
Name = Windows Visa XP # Name of OS
Description = 06.02.2007, 10:10 Es bootet. # Descriptive Text
Version = 1             # Version/Date of OS (optional)
Image = xp-20070727.rsync # Filename of rsync batch
BaseImage = xp.cloop # Filename of base partition image
Boot = /dev/hda1        # Partition containing Kernel & Initrd
Root = /dev/hda1        # root=/dev/partition Parameter (Root FS)
Kernel = grub.exe       # Relative filename of Kernel or Boot image
Initrd =                # Relative filename of Initrd
Append = --config-file=map(rd) (hd0,0); map --hook; \
          chainloader (hd0,0)+1; rootnoverify(hd0,0) \
          --device-map=(hd0) /dev/hda # grub.exe cmdline
StartEnabled = yes     # Enable "Start" Button
SyncEnabled = yes     # Enable "Synchronize" Button
RemoteSyncEnabled = yes # Enable "Synchronize from Server" Button

```

Im **[OS]**-Abschnitt dürfen Namen von Betriebssystemen mehrfach genannt werden, wobei die nachfolgenden Einstellungen und inkrementellen Image-Namen dann als Versionspaket dieses Betriebssystems interpretiert werden, und in den einzelnen Reitern für die Betriebssysteme im **linbo_gui** als „Subversionen“ auftauchen.

Die sehr lange **Append**-Zeile für den grub.exe-Bootlader wurde hier der Übersichtlichkeit halber mit \ und Zeilenumbrüchen wiedergegeben, dies ist jedoch in der **start.conf**-Konfigurationsdatei selbst nicht möglich. Jede Konfigurationsoption muss in einer Zeile für sich stehen, ohne Zeilenumbrüche!

4.1.2 PXE-Konfiguration (DHCP-Server)

LINBO-Kernel und LINBO-Dateisystem (**linbofs**) müssen sich in einem per TFTP erreichbaren Verzeichnis auf dem Server befinden. Ein klassischer Name für dieses Verzeichnis ist auf vielen Unix-Systemen **tftpboot**. Der für LINBO empfohlene TFTP-Server **atftpd** könnte dementsprechend auf dem Server wie folgt gestartet werden:

```
sudo atftpd -daemon --port 69 --retry-timeout 10 \
    --mcast-port 1758 --mcast-addr 239.239.239.0-255 \
    --mcast-ttl 1 --maxthread 100 --verbose=5 \
    /tftpboot
```

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 **linbofs.gz** in der Konfiguration von **pxelinux.cfg/CLIENT-ADRESSE** angegeben werden, in denen sich jeweils eine andere **start.conf**-Konfigurationsdatei befinden kann.²

Beispiel für einen entsprechenden Abschnit 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. LINBO-Kern und Images müssen ebenfalls per TFTP erreichbar sein.

Hinweis: Üblicherweise fügen die Clients ein Pfad-Präfix / zum Dateinamen hinzu, daher sollte für Testzwecke mit

```
atftp -r /linbo -l linbo 10.0.2.2
```

getestet werden, ob der LINBO-Kern über den TFTP-Server erreichbar ist, zumal ohne führenden / der TFTP-Server mitunter mit einem Fehler antwortet, statt die im TFTP-Exportverzeichnis liegenden Dateien zu liefern. V.a. der qemu-interne TFTP-Server zeigt dieses Verhalten.

²Später wird diese Konfiguration ein graphisches Verwaltungstool mit übernehmen helfen.

4.1.3 RSYNC-Konfiguration (Server)

Zur Synchronisation von Images sowie zum Upload neu erzeugter Images wird **rsync** (s. Abschnitt 1.3.5) verwendet.

Unter Debian wird rsync installiert mit **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 = /home/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 **/home/linbo** für alle Clients ohne Passwort. Mit

rsync server-adresse::linbo

können Sie das Verzeichnis testweise per rsync auflisten lassen, ohne eine Datei übertragen zu müssen.

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

```
[linbo-upload]
comment = LINBO Upload directory
path = /home/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 **/home/linbo**. Dort sollten sich der Linbo-Kernel **linbo** und das Linbo-Dateisystem **linbofs.gz** befinden, sowie die Image-Dateien mit den Betriebssystemen, Partitionsdumps und inkrementelle Änderungen (Abschnitt 1.3.5), für die Clients. Mit (Beispiel)

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

können Sie eine Testdatei (datei.txt) an den Server übertragen (allerdings klappt dies erst nach dem nächsten Konfigurationsschritt). Hierbei sollten Sie nach einem Passwort gefragt werden, was auch der Authentifizierung in LINBO dient.

Dieses Login/Passwort-Paar für die rsync-Freigabe **linbo-update** muss noch eingetragen werden, in die oben angegebene Datei **/etc/rsyncd.secrets**. Beispiel:

```
linbo:test
```

Der Benutzername „**linbo**“ ist momentan vom LINBO-System vorgegeben, das Passwort (hier: **test**) können Sie frei wählen. 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 **chmod 400 /etc/rsyncd.secrets** (als Administrator) 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 (**linbo**) und das LINBO-Dateisystem (**linbofs.gz**), sowie für das Booten von Windows, **grub.exe** in das LINBO-Verzeichnis (in unserem Beispiel **/home/linbo**) kopiert werden.

Damit LINBO diese Daten nicht jedesmal erneut herunterlädt, sollten auch die zu den genannten Dateien passenden **.info**-Dateien kopiert werden.

```
[grub.exe]
timestamp=200707251505
imagesize=198533
```

Abbildung 8: Beispiel für grub.exe.info

In diesen ist ein Zeitstempel und die Dateigrößen der Dateien vermerkt, so dass der 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 wirklich großen Image-Dateien angewandt. Hier erzeugt allerdings LINBO automatisch die entsprechenden info-Dateien und lädt sie mit hoch.

```
for i in linbo linbo.info linbofs.gz linbofs.gz.info \
    grub.exe grub.exe.info; do
    cp /home/development/LINBO/Images/$i /home/linbo/
done
```

Abbildung 9: Kopieren der LINBO-Dateien ins rsync-Repository

5 LINBO-Buildsystem

5.1 Systemvoraussetzungen

1. Installiertes POSIX-konformes Betriebssystem mit Bourne-kompatibler Shell (z.B. Debian „etch“). Cygwin sollte auch evtl. auch funktionieren, mit Linux-Crosscompiler.
2. GNU-Tar (zum Entpacken das Archives)
3. GNU-Make
4. GNU C-Compiler Version 3 oder höher (gcc-3.4 für qemu)
5. GNU-Binutils (zum Compilieren verschiedener Kernel-Komponenten notwendig)
6. Root-Rechte sind zum Bauen von LINBO NICHT erforderlich. Das Kernel-Buildsystem sorgt dafür, dass die Dateien im initramfs die erforderlichen Rechte erhalten, und dass auch Device-Dateien korrekt angelegt werden, daher kann als normaler User am System gearbeitet werden.
7. Zum Testen/Debuggen: qemu Version 0.9.0+cvs oder höher (-bootp Option erforderlich für simulierten PXE-Boot).

Der Bau von LINBO wird durch ein Makefile im LINBO-Verzeichnis gesteuert. **make** ohne Parameter liefert eine Kurzhilfe. Die einzelnen Schritte des Bauvorgangs sind recht selbsterklärend.

5.2 Verzeichnisse

Binaries enthält statische Binaries sowie dynamische Executables und Libraries für das initramfs. Die **Binaries/*.sh**-Dateien sind Shellskripte, die in LINBO den Bootvorgang und das GUI steuern.

Die für LINBO benötigten Dateien und Libraries werden in den Dateien **Kernel/initramfs_kernel.d/*.conf** sowie **Kernel/initramfs.d/*.conf** verwaltet. Dort sind auch neu hinzugefügte Dateien einzutragen, wenn sie in das initramfs aufgenommen werden sollen.

Das Verzeichnis **Graphics** enthält die Quellen des LINBO-Logos **linbo.xpm**, das im GUI dargestellt wird, sowie den Desktop-Hintergrund und das PXE-Bootbild

für LINBO. Diese Dateien werden nicht direkt in **Graphics** verwendet, sondern müssen bei Bedarf nach **Images** kopiert werden.

GUI enthält die Sourcen für **linbo_gui**, LINBOs graphische Oberfläche, sowie embedded Qt als Abhängigkeit (aus Platzgründen nicht im Repository).

Documentation enthält das Benutzer- und Administrationshandbuch von LINBO (u.a. dieses Dokument).

Kernel enthält den für LINBO verwendeten Linux-Kernel-Source. Wenn dieser aktualisiert wird, sollte die alte .config-Datei weiterverwendet werden, da sie die für das initramfs notwendigen Einstellungen enthält.

Images enthält den fertig gebauten LINBO-Kernel **linbo** als Hardlink auf Kernel/linux-*/arch/i386/boot/bzImage, **linbofs.gz** als zusätzliches initramfs mit den größeren Dateien, sowie den PXE-Bootlader **pxelinux.0**, Images und Archive zur Installation von LINBO und den gewünschten Betriebssystemen auf den Clients.

Sources ist ein Archiv der für die gebauten Binaries verwendeten Quelltexte, um die Binaries selbst neu bauen zu können, und die GNU GENERAL PUBLIC LICENSE §3 zu erfüllen. Für das Buildsystem ist das Verzeichnis eigentlich irrelevant, da es im Makefile nicht verwendet wird. Neu integrierte Programme sollten jedoch gewissenhaft in Sources archiviert werden (Debian: **cd Sources ; apt-get source paketname**).

5.3 Bauvorgang

Durch „**make**“ ohne Parameter dokumentiert:

```
WELCOME TO THE LINBO BUILD SYSTEM
```

```
make kernel (Re-)Build Kernel and Modules (recommended before "make linbo")
make linbofs (Re-)Build LINBO-FS
make linbo (Re-)Build LINBO-Kernel and LINBO-FS
make config Configure LINBO kernel and edit LINBO filesystem.
make clean Cleanup LINBO kernel source for recompilation.
make test Run LINBO kernel in qemu
make hptest Run LINBO from a harddisk-installed LINBO session
make pxetest Run LINBO in a qemu simulated PXE network boot
               (qemu 0.9.0+cvcs version required that supports '-bootp')
```

Don't worry about the sequence of build commands, this Makefile will tell you what to do first, in case anything is missing.

Have a lot of fun. ;-)

6 Das LINBO-Kochbuch, „How do I ...?“

6.1 Der allererste Start: Wie richte ich ein Master-System für LINBO-Images ein?

Grundsätzlich ist es praktisch, wenn das Master-System mit Hilfe von LINBO partitioniert wurde, damit die Partitionsgrößen und -namen zuverlässig festgelegt sind. Siehe Rezept 6.2.

```
[LINBO]
Cache = /dev/hda2
Server = 10.0.2.2

[Partition]
Dev = /dev/hda1          # Windows-Partition
Size = 4000000
Id = 7
FSType = ntfs
Bootable = yes

[Partition]
Dev = /dev/hda2          # Cache-Partition
Size = 4000000
Id = 83
FSType = reiserfs
Bootable = no

[OS]
Name = Win
Description = Installation vom 06.02.2007
BaseImage = win.cloop    # Komplette Partition (Basis)
Image = win-20070727.rsnc # Inkrementelles Update
Boot = /dev/hda1
Root = /dev/hda1
Kernel = grub.exe
Initrd =
Append = --config-file=map(rd) (hd0,0); map --hook; ... (s.o.)
StartEnabled = yes
SyncEnabled = yes
RemoteSyncEnabled = yes
```

Mit einer minimalen **start.conf** wir der hier gezeigten könnte beispielsweise begonnen werden. Bitte beachten Sie, dass für LINBO eine Cache-Partition (hier: /dev/hda2) eingerichtet werden muss, die groß genug ist, um alle Betriebssysteme vorzuhalten, die auf den Clients installiert werden sollten (plus etwas Platz für vielleicht einmal selbsterzeugte Voll- und Inkrementalimages). Siehe Rezept 6.3.

Die etwas längliche **grub.exe**-Zeile unter **Append** ist hier nur unvollständig wiedergegeben (s.a. Abschnitt 4.1.1 auf S. 15).

Nach der Partitionierung durch LINBO kann das gewünschte Betriebssystem auf der (in diese Beispiel) ersten Partition mit einer Installations-CD eingerichtet wer-

den, und anschließend mit LINBO in ein Image (Basisimage win.cloop, spätere Änderungen in win-20070727.rsinc) umgewandelt und zum LINBO-Server übertragen werden.

6.2 Was muss in der **start.conf** stehen?

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

Die Cache-Partition hält eine Kopie der Installations-Images für jedes Betriebssystem, das auf den Clients bei Bedarf neu aufgesetzt, synchronisiert oder aktualisiert werden soll. Grundsätzlich ist es LINBO egal, auf welcher Partitionsnummer diese Partition liegen muss, bzw. ob es eine „primäre“ (/dev/hda1...hda4) oder „logische“ (/dev/hda5...∞) Partition (Festplatten-Jargon) ist. Der Name der Partition muss in der **start.conf** unter Abschnitt **[LINBO]**, Parametername **Cache** eingetragen werden, und die Größe in einem Abschnitt **[Partition]** so wie in diesem Beispiel:

```
[LINBO]
Server = 10.0.2.2
Cache = /dev/hda2  # <- Das ist die Cache-Partition

[Partition]
Dev = /dev/hda2      # Name
Size = 20000000      # Größe in kB
Id = 83              # Partitionstyp (83 = Linux)
FSType = reiserfs    # Dateisystem
Bootable = no        # Egal
```

In diesem Beispiel (die Kommentare mit # sind optional) wird eine 20 GB große Cache-Partition verwendet. Der Device-Name dieser Partition, **/dev/hda2** ergibt sich aus der Tabelle im Rezept [6.4](#).

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

[TODO: Rezepte und Screenshots der aktuellen Version beifügen.]

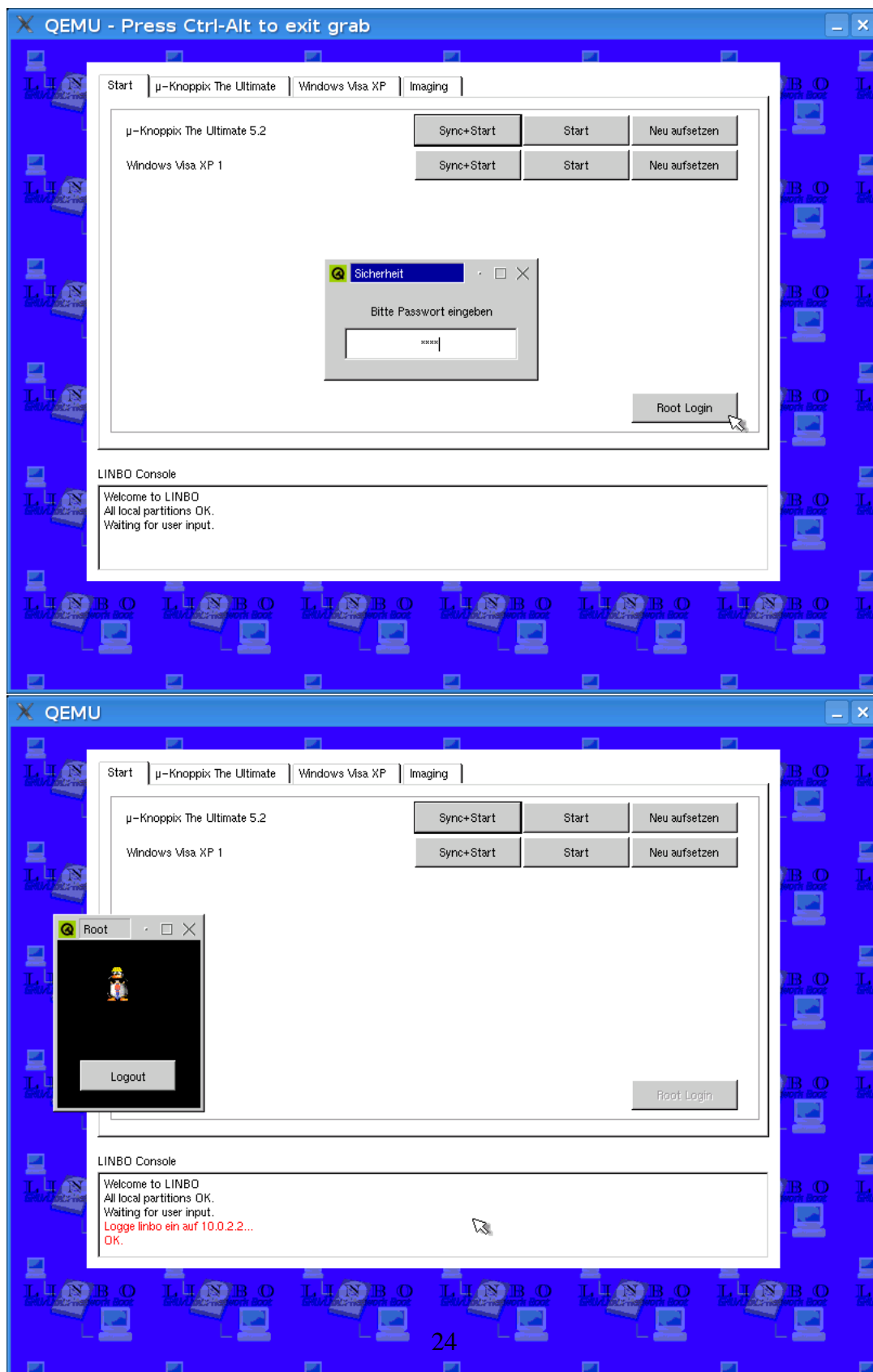


Abbildung 10: Einloggen als Admin bei LINBO

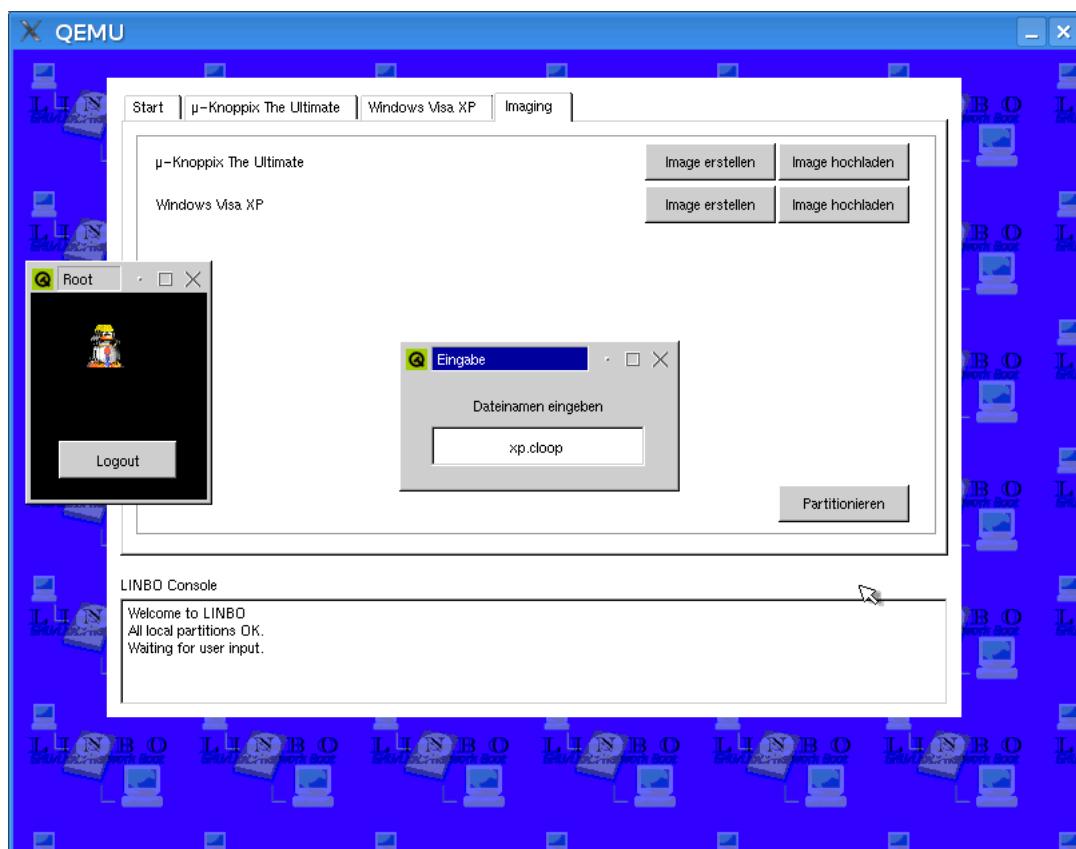


Abbildung 11: Image von Partition erzeugen, Dateiname

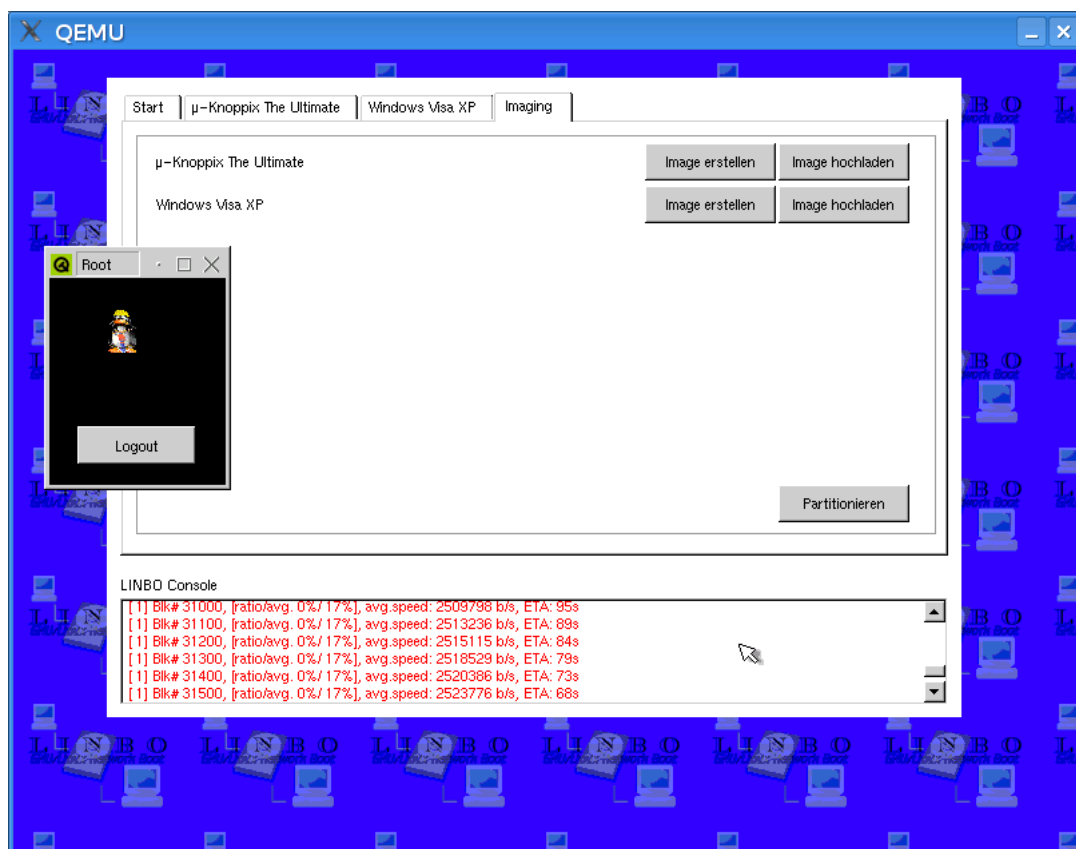


Abbildung 12: Image von Partition erzeugen, Kompression

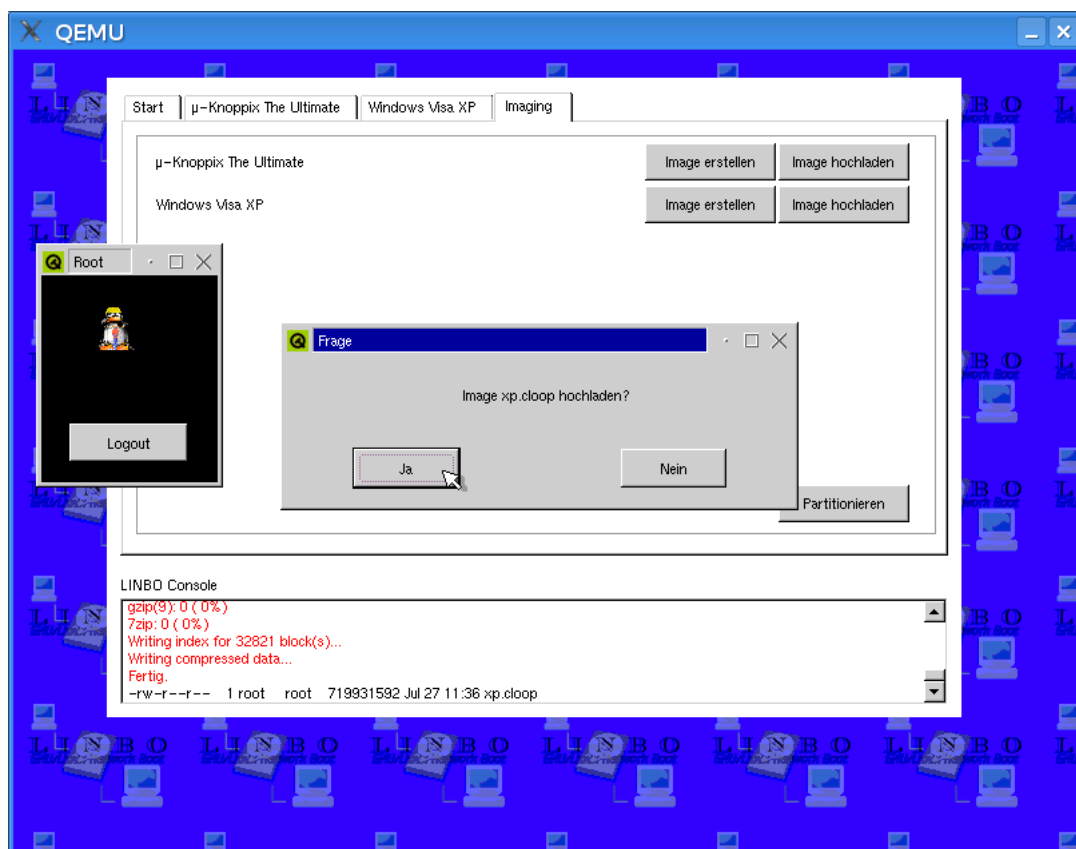


Abbildung 13: Image von Partition erzeugen, Upload

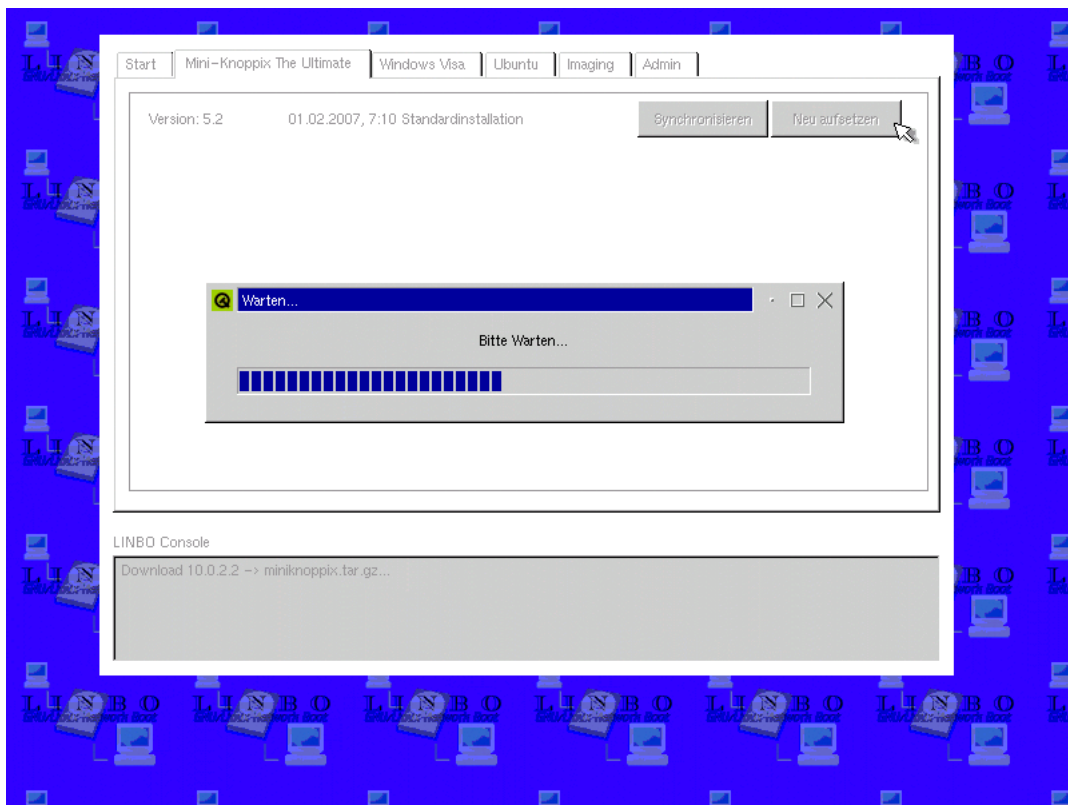


Abbildung 14: Neu aufsetzen (1)

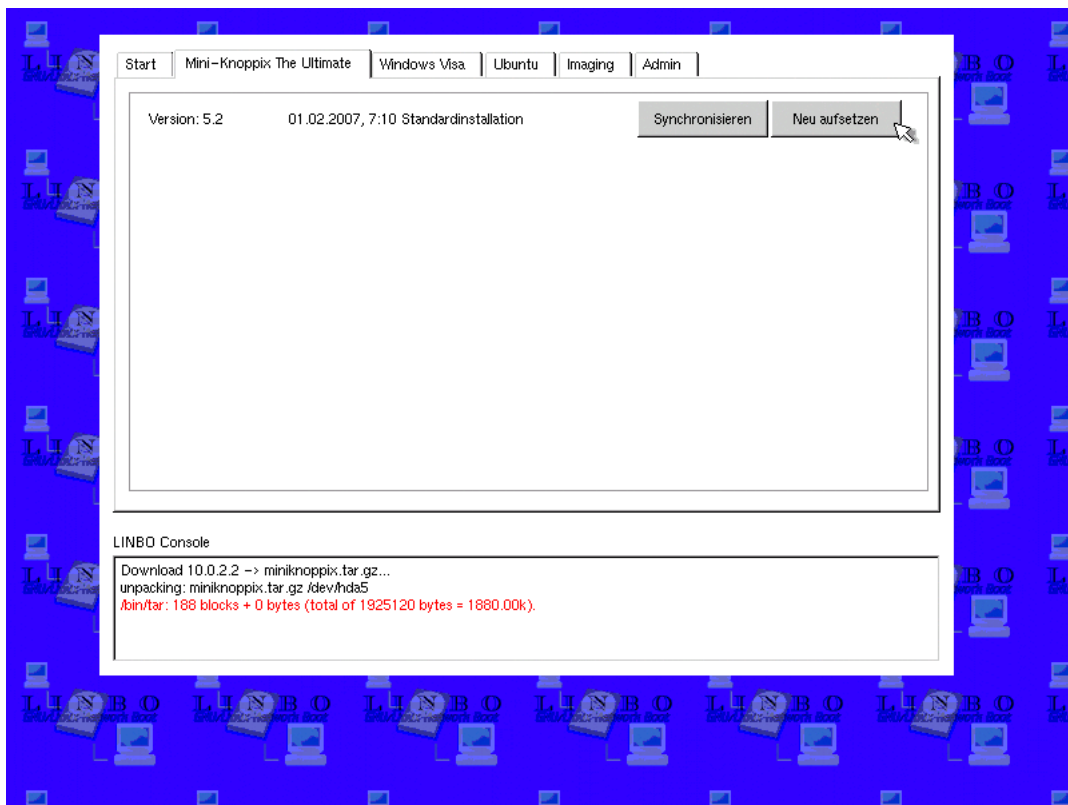


Abbildung 15: Neu aufsetzen (2)

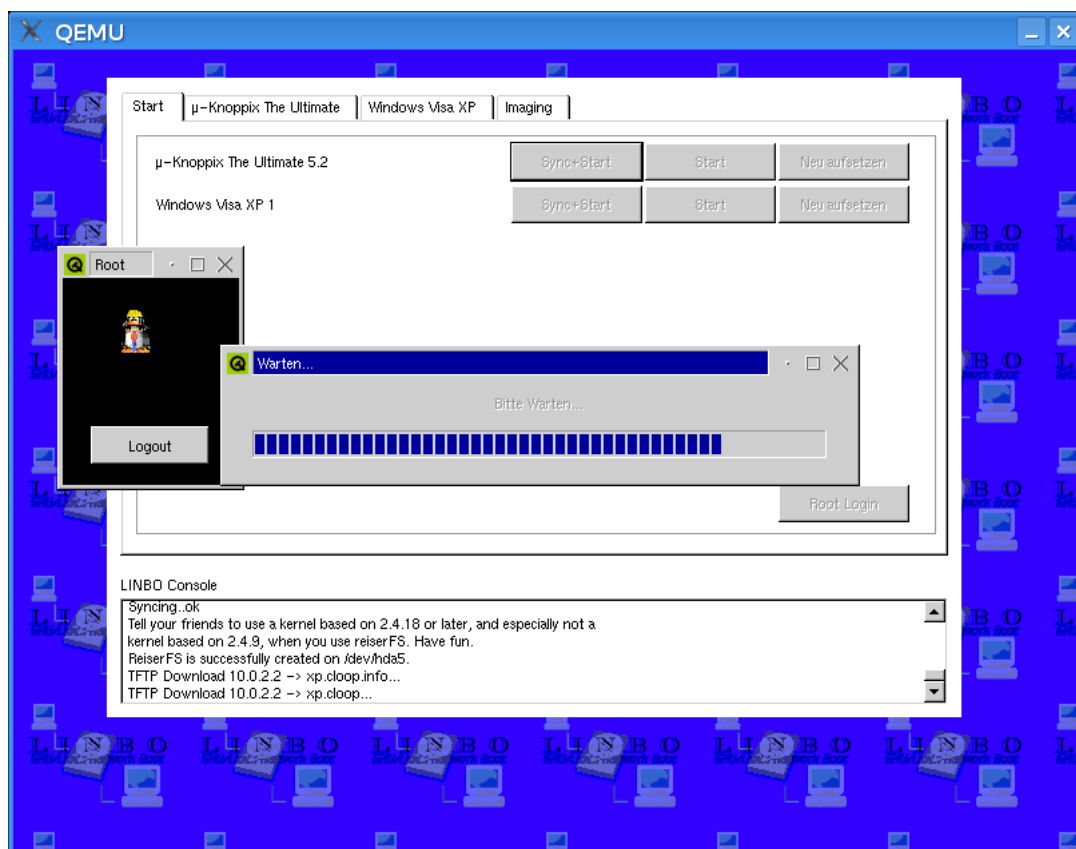



Abbildung 16: Sync+Start



```
ide0: BM-DMA at 0xc000-0xc007, BIOS settings: hda:pio, hdb:pio
ide1: BM-DMA at 0xc008-0xc00f, BIOS settings: hdc:pio, hdd:pio
hda: QEMU HARDDISK, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hdc: QEMU CD-ROM, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
hda: max request size: 1024KiB
hda: 8388608 sectors (4294 MB) w/256KiB Cache, CHS=8322/255/63
hda: cache flushes supported
hda: hda1 hda2 hda3 hda4 < hda5 hda6 >
hdc: ATAPI 4X CD-ROM drive, 512kB Cache
Uniform CD-ROM driver Revision: 3.20
mice: PS/2 mouse device common for all mice
input: AT Translated Set 2 keyboard as /class/input/input0
EISA: Probing bus 0 at eisa.0
EISA: Detected 0 cards.
NET: Registered protocol family 2
input: ImExPS/2 Generic Explorer Mouse as /class/input/input1
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 98304 bytes)
TCP bind hash table entries: 8192 (order: 4, 98304 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
NET: Registered protocol family 1
NET: Registered protocol family 15
Using IPI No-Shortcut mode
ACPI wakeup devices:
ACPI: (supports S5)
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 296k freed
#
```

Abbildung 17: OS-Boot

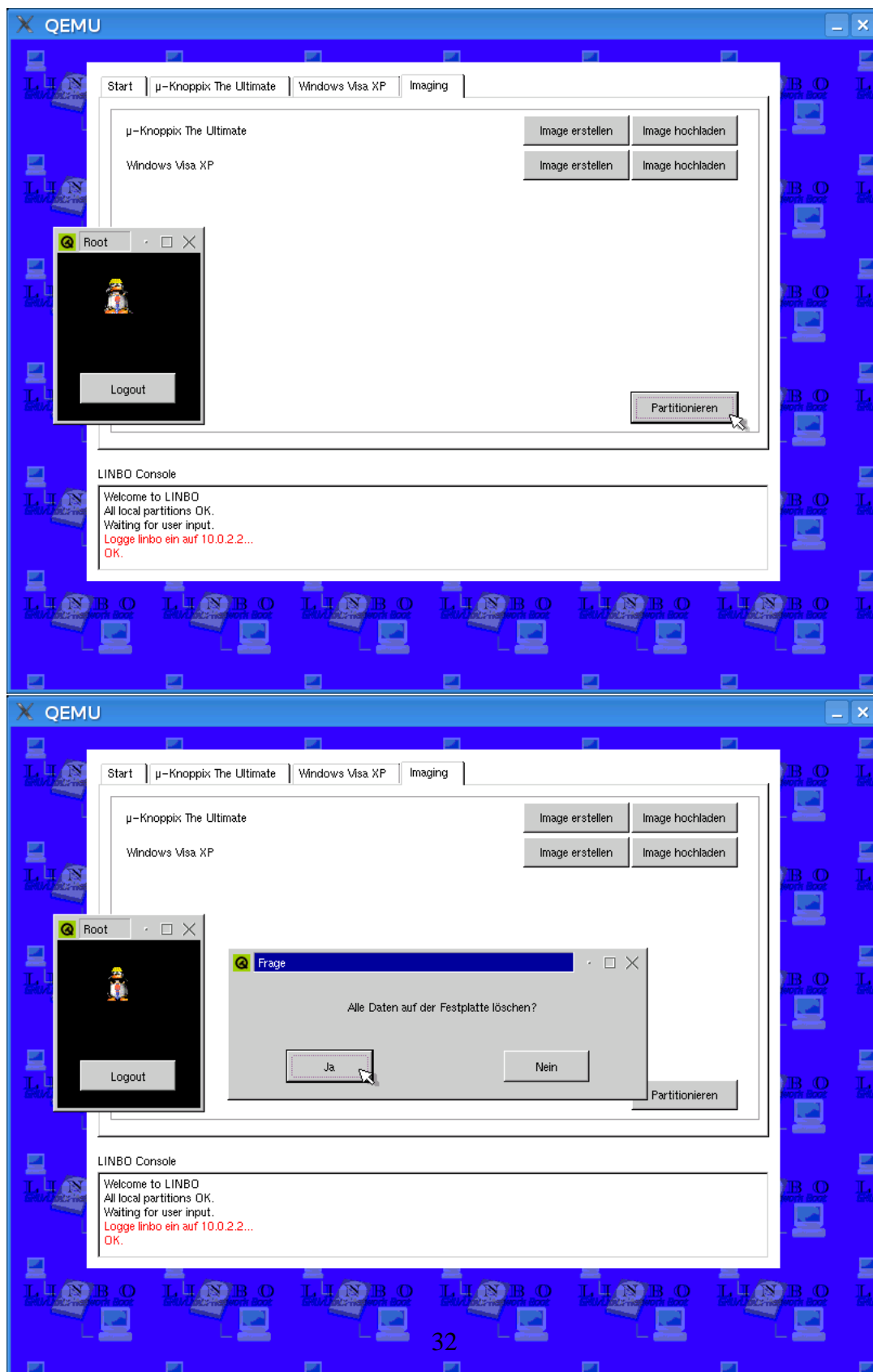


Abbildung 18: Partitionieren