

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

Table of Contents

OpenSLX	1
Wiki.....	1
Schritt-für-Schritt-Anleitung: Stateless Linux mit OpenSLX	2
Weitere Entwicklungen.....	3
Installation.....	3
Funktionsweise.....	4
Der SLX-Server.....	5
Programmnamen und globale Einstellungen.....	6
Stage1 aus Netzwerk-Quellen oder Referenzmaschine.....	7
Plugins - die modularen Erweiterungen des Grundpakets.....	9
Systeme exportieren.....	9
Quickstart.....	10
Die Datenbank und das Boot-Setup.....	11
Unterhaltung mit der Datenbank.....	12
System-Update und Ergänzungen.....	13
Aufräumarbeiten.....	14
Bootablauf.....	14
PXE-Boot.....	14
DHCP für die IP-Adressverwaltung.....	15
TFTP.....	15
Syslinux.....	16
Konfiguration von OpenSLX Clients.....	17
Die Kernel Commandline.....	18
Zentrale Konfigurationsdatei machine-setup.....	19
Probleme.....	19

OpenSLX

- [Overview](#)
- [Activity](#)
- [Roadmap](#)
- [Issues](#)
- [Wiki](#)
- [Repository](#)

Wiki

[Start page](#)

[Index by title](#)

[Index by date](#)

[History](#)

Schritt-für-Schritt-Anleitung: Stateless Linux mit OpenSLX

(Fassung muss überarbeitet werden + Links auf Wikis, Generierung einer ASCII-Fassung als README für das OpenSLX-Basispaket)

Diese Anleitung dient als Schnelleinstieg in die Erstellung einer Umgebung aus Linux diskless/stateless Workstations. Sie geht deshalb nicht auf die Tiefen der Konfiguration ein, sondern zeigt Beispiele, wie ein Standard-Setup aussehen könnte.

Der Betrieb und die Administration jeder größeren Anzahl von Rechnern ist mit Aufwand verbunden. Dieses gilt ebenso für Linux-Installationen. Damit bei Ihnen die Einrichtung und Wartung großer Installationen nicht zur Hauptaufgabe Ihrer Administratoren wird, bedarf es spezieller, auf diese konkrete Aufgabe zugeschnittener Technologien, um diese Probleme in den Griff zu bekommen. Neben den klassischen Verfahren des Software-Deployments und des Patch-Managements, haben Stateless-Ansätze beste Chancen den Admin von langweiligen Routineaufgaben zu entlasten. Je nach Anforderung kann eine einzige Installation quasi beliebig viele Client-Maschinen über das Netz bedienen. Dazu bedarf es keiner speziellen Vorbereitung der Maschinen: PXE oder Etherboot reichen aus, um das Gerät aus dem Netzwerk booten zu lassen. Hier besteht natürlich zusätzlicher Gestaltungsspielraum: Lokale Installationen müssen nicht weichen, sondern können, beispielsweise in Migrations- oder Mischszenarien, alternativ zum Stateless-Betrieb weiterhin angeboten werden.

Bei OpenSLX handelt es sich um GPL-Software, die es erlaubt festplattenlose Linux-Workstations einfach zu installieren und leicht zu administrieren. Das Projekt fokussiert auf große Installationen mit recht ähnlichem Einsatz-Profil (also nicht unbedingt Rechner mit Spezialhardware zur Produktionssteuerung o.ä.). So wird es derzeit in Schulen und Universitäten eingesetzt, um hochflexible und stabile Poolumgebungen für Schulungen, Unterricht, freies Arbeiten etc. zu schaffen. Durch Virtualisierungstechniken (derzeit VMware, später auch VirtualBox oder Xen) kann auf der geschaffenen Stateless-Linux-Basis ein weiteres OS zu Schulungszwecken, die Windows- oder Linux effektiv verteilt betrieben werden.

Die Umschaltung zwischen verschiedenen Betriebsmodi erfolgt zur Bootzeit, da keine aufwändige und zeitraubende Imageverteilung vor (Festplattenaustausch) oder während des Rechnerstarts (Imagecopy) erfolgt.

Gute Linux-Distributionen gibt es genug: Deshalb wählt OpenSLX nicht den Ansatz, eine eigene Paketsammlung anzubieten, sondern stellt eine Skriptsammlung zur Verfügung, welche viele Standard-Distribution für den Stateless-Betrieb vorbereitet. Das OpenSLX-Paket stellt Ihnen eine Middleware bereit, die für den Benutzer einer Maschine optimalerweise unsichtbar bleibt. Es besteht das jeweilige typische Look & Feel der gewählten Distribution, die Anpassungen und Änderungen finden im Hintergrund statt.

Anwender greifen an ihrem Desktop wie gewohnt auf alle an der Maschine verfügbaren Geräte zu: Es gibt keine speziellen Herausforderungen, da alles lokal geschieht und nicht aufwändig beispielsweise mittels Automount-Ketten über das Netzwerk erfolgen muss. Das Abspielen von Filmen oder die Nutzung von 3D-Anwendungen sind deshalb ebenfalls kein Problem. Zudem kann nun so das umfangreiche Leistungspotential heutiger Arbeitsplatz-PCs vollständig genutzt und dem Benutzer exklusiv zur Verfügung gestellt werden.

Der auf diese Weise angebotene Linux-Desktop lässt sich in gewohnter Weise für vielfältige Anwendungen nutzen: So können in Virtualisierungsumgebungen wie VirtualBox oder VMware (Workstation und Player) weitere, auch proprietäre Betriebssysteme gebootet werden.

Weitere Entwicklungen

Mit Techniken wie UnionFS oder AUFS können bestehende Basisinstallationen leicht erweitert oder verschlankt werden. So lassen sich unterschiedliche Nutzergruppen verschieden bedienen. Zudem wird es möglich, dass eine Zentrale die Grundinstallation/Infrastruktur bereitstellt und dezentral weitere Software einfach in das System integriert wird.

Ganz andere Möglichkeiten bieten sich durch virtuelle Workstations: Statt direkt auf der jeweiligen Maschine die Installation des Betriebssystems für einen bestimmten Nutzer vorzunehmen, erfolgt sie in eine virtuelle Umgebung, deren Container im Homeverzeichnis des Benutzers abgelegt wird. So können Benutzer sich an einer beliebigen, in einem OpenSLX Verbund betriebenen Maschine anmelden und finden dort ihre gewohnte Umgebung vor.

Installation

Das OpenSLX-Paket ist, wie eigentlich jedes Opensource-Paket, nicht fertig, sondern wird beständig seit 1997 weiterentwickelt. Aktuell ist derzeit die Version 4.9. Sie können sie von der Projekt-Homepage als RPM- oder Debian-Paket herunterladen (nicht wirklich); für OpenSUSE wählen Sie das RPM:

```
x60s:~ # wget
x60s:~ # rpm -i
```

Für aktuelle Bugfixes, neue Features, wie weitere unterstützte Distributionen, können Sie jederzeit anonym auf die Quellen im Subversion zugreifen:

```
x60s:~ # svn co http://svn.openslx.org/svn/openslx/openslx
Ausgecheckt, Revision 2NNN.
x60s:~ # cd openslx/trunk
x60s:~/openslx/trunk # make install
```

In den meisten Fällen müssen noch einige Perl-Pakete hinzugefügt werden: Unter Ubuntu heißen diese libclone-perl, libdigest-sha1-perl, libconfig-perl, libdbi-perl, libdbd-mysql-perl und libdbd-sqlite3-perl.

Unabhängig vom Weg, wie Sie OpenSLX installiert haben, arbeitet OpenSLX in seiner Grundeinstellung mit folgender Verzeichnisstruktur:

```
/etc/opt/openslx
/etc/opt/openslx/distro-info
/opt/openslx
/opt/openslx/lib
/opt/openslx/share
/opt/openslx/bin
/srv/openslx
/srv/openslx/export
/srv/openslx/tftpboot
/var/opt/openslx
/var/opt/openslx/config
/var/opt/openslx/db
/var/opt/openslx/stage1
```

Nur die Verzeichnisse mit den statischen SLX-Dateien werden angelegt, Verzeichnisse wie `/var/opt/openslx` oder `/srv/openslx` erst beim Setup von Client-Umgebungen. Das Kommando `slxsettings` erlaubt Ihnen Anpassungen:

```
x60s:~ # slxsettings set private-path='/var/opt/openslx' set public-path='/'
setting private-path to '/var/opt/openslx'
setting public-path to '/'
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

Unterhalb von `/etc/opt/openslx` landen alle globalen Einstellungen des Pakets. In `/opt/openslx` liegen alle statischen Dateien, wie Skripten und die dafür benötigten Perl-Module und Includes. In `/var/opt/openslx`, wenn nicht durch `slxsettings` anders festgelegt, landen alle Daten, die zur Laufzeit generiert werden. Die Verzeichnisse sind nach der Installation noch leer. In `config` können Sie Dateien und Konfigurationen zu den später exportierten Systemen ablegen, `db` nimmt die Datenbank Ihrer Systeme und Clients auf.

Die in einem weiteren Schritt zu exportierenden Systeme landen schlussendlich in Stage1. Während die Datenmengen in den anderen Verzeichnissen zu vernachlässigen sind, sollten Sie hier je nach Art und Zahl der OpenSLX-Installationen (Stage1) mindestens 5 - 50 GByte vorsehen.

Sollten die Festlegung der verschiedenen Verzeichnisse nicht Ihren Vorstellungen entsprechen, können Sie diese mit dem Skript `slxsettings` anpassen. So können Sie beispielsweise dafür sorgen, dass die Stage1-Daten nicht in Ihrem Server-Rootfilesystem sondern beispielsweise auf einer separat gemounteten Datenpartition zu liegen kommen.

Die aktuelle Version Ihres OpenSLX-Toolsets liefert folgender Aufruf:

```
x60s:~ # slxversion
5.0.0revNNNN
```

Funktionsweise

Die Aufgabe des OpenSLX-Projekts ist die Einrichtung von exportierbaren Systemen; Distributionen in einer bestimmten Version, die von stateless Clients über das Netzwerk als Rootfilesystem eingebunden werden können. Dabei kann ein SLX-Server durchaus mehrere solcher Systeme auf verschiedene Weise bereitstellen.

Für eine gute Übersichtlichkeit ist die Setup-Prozedur für Ihre stateless Clients in mehrere klare Stadien aufgeteilt:

- **PreBoot** steht für ein Mini-Linux-System, welches als Ersatz-Boot-Umgebung in PXE-losen Szenarien zum Einsatz kommen kann. Eine weitergehende Beschreibung findet sich [hier](#).
- **Stage0** bezeichnet ein laufendes Referenzsystem auf realer oder virtueller Hardware, von dem durch Rsync ein Abzug auf den Server generiert wird (**Stage1**). Dieses Stadium spielt lediglich für den Klonmodus eine Rolle.
- **Stage2** - aus dem Stage1 können verschiedene Client-Rootfilesysteme erzeugt werden. Dazu wird das Stage1 geeignet dupliziert, d.h. unter Weglassung für den stateless Betrieb irrelevanter Dateien, eine neues Unterverzeichnis im NFS-Export erzeugt oder für Netzwerk-Blockdevices ein komprimiertes SquashFS angelegt. Dazu gehört mindestens ein passender Kernel (üblicherweise direkt aus der verwendeten Distribution), mindestens ein zum Kernel passendes InitialRamFS und eine Client-Konfiguration. Die Client-Konfiguration kann eine Reihe von Dateien enthalten, die dann im Stage3 an die passende Stelle entpackt werden.
- **Stage3** - Während Stage0-2 administrative Vorgänge auf dem SLX-Server umfassen, die Sie nur wenige Male ausführen müssen, findet Stage3 auf jedem einzelnen Client bei jedem Neustart innerhalb des InitialRamFS statt. Hier erfolgt das individuelle Setup der Maschine. Diesen Bereich decken im Großen und Ganzen die Shell-Skripten `init` (SLX-Init), `hwautocfg` und `servconfig` sowie die individuelle Konfiguration von Plugins ab. `serverconfig` legt alle wichtigen Konfigurationsdateien des Clients in `/mnt/etc` an oder modifiziert diese. Lokale Erweiterungen (für alle Clients, die ein bestimmtes InitRamFS verwenden) sind mittels `preinit.local`, ganz am Anfang von SLX-Init, und `postinit.local` ziemlich zum Ende von SLX-Init möglich. `postinit.local` lässt sich mittels ConfTGZ (Client-Konfiguration) zur Verfügung stellen. Das

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

ist Paket von Konfigurationsdateien, die im Stage3 beispielsweise per TFTP gezogen und anschliessend entpackt werden. Die Vorlage hierzu liegt in `/var/opt/openslx/config/<system>. <system>` ist dabei der Name des Vendor-OS verknüpft durch zwei Doppelpunkte mit der Art des Exports.

- In **Stage4** finden aus SLX-Sicht nur noch kleinere Konfigurationsaufgaben statt. Nun ist der Client mit einem kompletten Rootfilesystem, allen Programmen und Dateien ausgestattet, so dass das Setzen der Tastatureinstellungen erfolgen kann. Ebenso wird in diesem Stadium der grafische Login zu einem geeignet frühen Zeitpunkt gestartet.

Je nach Art der Clients wollen Sie auf einem Server eine oder mehrere verschiedene System-Exporte bereitstellen. Das kann einerseits bedeuten, dass ein Stage1-System in mehreren Varianten, wie NFS und gleichzeitig NBD/SquashFS, an die Clients ausgeliefert werden soll. Andererseits können auch sehr unterschiedliche Stage1 mit der selben Technologie aus unterschiedlichen Exports bedient werden. Das ist beispielsweise für on-site Tests sinnvoll, wenn man neben dem aktuell benutzten System schon das neue System parallel anbieten möchte.

Möchte man ein und dasselbe Grundsystem in verschiedenem Umfang exportieren, gibts auch hier zwei Möglichkeiten: Entweder Sie legen von vorneherein zwei unterschiedliche Stage1-Varianten an, die separat gepflegt werden. Oder Sie verschieben die Differenzen in einen speziellen Bereich, der in einer der Umgebungen später zusätzlich gemountet wird beispielsweise unter Verwendung von Stacking-Filesystemen.

Der SLX-Server

Alle eben gemachten Überlegungen haben Auswirkungen auf den Platzbedarf. Je nach Umfang des installierten Systems benötigt eine Stage1-Variante 2-8GByte Plattenplatz im Bereich `/var/opt/openslx/stage1`. Ein Stage2-NFS-Export benötigt zwischen 90 und 98% dieser Menge, ein NBD/SquashFS-Export zwischen 35 und 50%. Entsprechend sollten Sie auch `/srv/openslx/export` mit Plattenplatz ausstatten.

SLX-Clients arbeiten stateless und haben lokal kein Betriebssystem installiert. Sie beziehen ihr gemeinsames Rootfilesystem von einem zentralen Server, der entsprechend leistungsfähig ausgelegt sein sollte. Die größte Last trägt der das Rootfilesystem exportierende Dienst, der den höchsten Dateisystem-Output generiert. Für ein Setup bestehend aus 100 Clients sollten Sie Ihrem Server ein bis zwei Gigabit-Interfaces und schnelle Festplatten im Raid-Verbund spendieren. Da jedoch nur die Daten im Bereich des Exports von den Clients eingebunden werden, können Sie `/var/opt/openslx/stage1` auf langsame Platten legen. Die Anforderungen von DHCP und TFTP überfordern einen durchschnittlichen Server auch nicht bei mehr als 1000 Clients. Während der Server besser am Backbone Ihres Netzwerkes hängt, genügt für die Clients ein 100 Mbit/s Interface. Je nach Art der Sitzung werden im Laufe eines Tages zwischen 0,3 und 3 Gigabyte an Daten pro Client transportiert. Der Einsatz eines Netzwerk-Blockdevices mit daraufliegendem SquashFS kann diese Datenmenge gegenüber NFS mehr als halbieren.

Eine Steigerung der Clientanschlüsse auf 1 Gigabit sollte mit verbessertem IO des Servers einhergehen: Ein sehr schnelles Raid oder Lage der Exports im RAM verbunden mit Gigabit-Trunking beschleunigen den Start und Betrieb der Clients erheblich. Sie sind dann üblicherweise fühlbar schneller als im klassischen Betrieb von lokaler Festplatte.

Für kleinere Netze oder Demonstrationen mit wenigen Clients genügen durchaus durchschnittlich ausgestattete Fileserver oder ein normaler Desktop-PC.

Programmnamen und globale Einstellungen

Die Konfigurations- und Setup-Programme tragen alle das Prefix "slx", die meisten davon sind Perl-Skripten. Eine Kurzhilfe liefern die meisten Tools mit der Option `--help`, die Manpage kann durch `--man` erreicht werden.

Nach der Installation können Sie mittels `slxsettings` globale Grundeinstellungen, wie Pfade, an Ihre Vorstellungen anpassen. Ohne Parameter aufgerufen, zeigt Ihnen `slxsettings` die Liste der Einstellungen:

```
x60s:~ # slxsettings
paths fixed at installation time:
    base-path='/opt/openslx'
    config-path='/etc/opt/openslx'
current base settings (cmdline options):
    db-name='openslx'
    db-spec=_
    db-type='SQLite'
    debug-confess=_
    locale='de_DE.UTF8'
    locale-charmap='UTF-8'
    private-path='/var/opt/openslx'
    public-path='/'
    temp-path='/tmp'
    verbose-level=_
extended settings:
    db-passwd=<unset>
    db-user=<unset>
    default-shell='bash'
    default-timezone='Europe/Berlin'
    mirrors-preferred-top-level-domain='de'
    mirrors-to-try-count='20'
    mirrors-to-use-count='5'
    ossetup-max-try-count='5'
    pxe-theme=<unset>
    pxe-theme-menu-margin='9'
```

`slxsettings` verfügt über eine umfangreiche Hilfe (`slxsettings --man`), die zudem für alle perl-basierten SLX-Kommandos die gemeinsamen Optionen zusammenfasst:

```
SLXSETTINGS(1)          User Contributed Perl Documentation          SLXSETTINGS(1)

NAME
    slxsettings - OpenSLX-script to show & change local settings

SYNOPSIS
    slxsettings [options] [action ...]

Script Actions

    set <option-name=value>    sets the option to the given value
    reset <option-name>        resets the given option to its default

List of Known Option Names

    db-name=<string>           name of database
    db-spec=<string>           full DBI-specification of database
    db-type=<string>           type of database to connect to
    locale=<string>           locale to use for translations
    log-level=<int>           level of logging verbosity (0-3)
    logfile=<string>          file to write logging output to
    private-path=<string>     path to private data
    public-path=<string>     path to public (client-accessible) data
    temp-path=<string>        path to temporary data
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

General Options

```
--help          brief help message
--man           full documentation
--quiet        do not print anything
--version      show version
```

Actions

```
set <openslx-option>=<value>
    sets the specified option to the given value

reset <setting>
    removes the given setting from the local settings (resets it to
    its default value)
```

DESCRIPTION

```
slxsettings can be used to show or change the local settings for
OpenSLX.
...
```

So lässt sich bei allen SLX-Kommandos die Debug-Ausgabe durch die Option `--log-level=0-3` steuern. Änderungen der Werte lassen sich mit `set` vornehmen, `reset` sorgt für ein Zurücksetzen auf den Default-Wert, so vorhanden, oder löscht den Eintrag.

Für die meisten SLX-Umgebungen sollten die Default-Einstellungen passen. Wenn Sie aber beispielsweise unter `/data/openslx` eine grosse leere Plattenpartition gemountet haben und diese für OpenSLX nutzen möchten, können Sie die Stage1-Installationen auf diese umbiegen:

```
x60s:~ # slxsettings set public-path='/data/openslx'
setting public-path to '/data/openslx'
x60s:~ # slxsettings set mirrors-preferred-top-level-domain=de
setting mirrors-preferred-top-level-domain to 'de'
```

Das zweite Beispiel zeigt eine wichtige Einstellung für die Suche von Spiegeln der Installationsquellen der verschiedenen Distributionen. OpenSLX versucht zwar die Domain bei der Installation zu ermitteln, wird aber gerade in Netzen mit privaten Domainnamen scheitern. Ebenso können Sie die Art der Datenbank zur Verwaltung der mit OpenSLX exportierten Systeme und Ihrer Clients ändern. Der Standardtyp ist "SQLite", je nach installierten Perl-Komponenten können Sie alternativ auch MySQL als Backend einsetzen.

```
x60s:~ # slxsettings set db-type=SQLite
setting db-type to 'SQLite'
```

Haben Sie sich einmal für einen Datenbanktyp entschieden, können Sie diesen im laufenden Betrieb nicht trivial ändern. Nach den Grundeinstellungen können Sie sich nun den Stadien 0 bzw. 1 widmen.

Die Einstellungen finden Sie in `/etc/opt/openslx/settings`:

```
SLX_DB_TYPE=SQLite
SLX_MIRRORS_PREFERRED_TOP_LEVEL_DOMAIN=de
SLX_PUBLIC_PATH=/data/openslx
```

Stage1 aus Netzwerk-Quellen oder Referenzmaschine

Je nach Linux-Distribution stehen Ihnen ein oder zwei Wege des Stage1-Setups zur Verfügung: Entweder Sie installieren ein ganz frisches System aus den Paketquellen oder Sie klonen eine schon fertig eingerichtete Maschine, die bereits unter Ihrer Wahldistribution läuft. Beides übernimmt das Kommando `slxos-setup`, welches den Aufbau `slxos-setup <parameter>` hat. Auskunft gibt das

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

Subkommando `list-supported`:

```
x60s:~ # slxos-setup list-supported
gentoo-2007.X          (clone)
suse-10.2              (clone,install)
suse-10.2_x86_64      (clone,install)
suse-10.3              (clone)
suse-10.3_x86_64      (clone)
suse-11.0              (clone,install)
suse-11.0_x86_64      (clone,install)
ubuntu-8.04            (clone,install)
ubuntu-8.04_amd64     (clone,install)
```

Bisher werden nicht alle aufgelisteten Distributionen vom SLX-Stage3 unterstützt. Neuinstallation nimmt `slxos-setup` mit dem Subkommando `install` vor, gefolgt von der gewünschten Distribution:

```
x60s:~ # slxos-setup install suse-11.0
[ ... lots of yum output ... ]
Total download size: 69 k
Downloading Packages:
(1/2): nbd-2.8.7-14.i586. 100% |=====| 45 kB    00:00
(2/2): squashfs-kmp-defau 100% |=====| 24 kB    00:00
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: squashfs-kmp-default          ##### [1/2]
  Installing: nbd                          ##### [2/2]
Installed: nbd.i586 0:2.8.7-14 squashfs-kmp-default.i586 0:3.1_2.6.18.2_34-34
Complete!
Starting SUSEconfig, the SUSE Configuration Tool...
Running in full featured mode.
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.perl...
Executing /sbin/conf.d/SuSEconfig.permissions...
Finished.
Vendor-OS 'suse-11.0' installed succesfully.
Vendor-OS 'suse-11.0' has been added to DB (ID=1).
```

Je nach Bandbreite und Auslastung Ihres Netzwerkes können Sie eine Kaffee- oder Mittagspause bis zum Abschluss dieses Schrittes einplanen. Jede so aufgesetzte Distribution können Sie später mit `slxos-setup update <distro>`; auffrischen, was dann deutlich schneller als die Primärinstallation geht.

Wenn Sie schon ein fertig eingerichtetes System haben, weil Sie beispielsweise bestehende Fat-Clients in OpenSLX-Clients umwandeln wollen, kann ein Klon-Setup eine sinnvolle Option sein. Das `slxos-setup` Subkommando `clone` benötigt dazu die Ausgangsquelle. Um ein geklontes System von einer klassischen RPM-Installation wie zuerst gezeigt unterscheiden zu können, geben Sie dem Kind am besten einen individuellen Namen. Alles, was Sie hinter dem Distributionsnamen und der Version angeben, im Beispiel "-mysystem" wird Bestandteil des Namens. Der eigentliche Klon-Vorgang erfolgt im Hintergrund mit dem Kommando `Rsync`, so dass Sie am Anfang nach dem Root-Passwort der Referenzmaschine gefragt werden. Auf einem Ubuntu-Referenzsystem müssen Sie dafür sorgen, dass "root" sich per SSH anmelden darf.

```
x60s:~ # slxos-setup clone 10.60.4.50:/ suse-11.0-mysystem
Cloning vendor-OS from '10.60.4.50:/' to '/space/stage1/suse-11.0-mysystem'...
Password:
receiving file list ...
...
var/yp/
var/yp/nicknames
var/yp/binding/

sent 5765235 bytes  received 6044617832 bytes  5710602.23 bytes/sec
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

```
total size is 6025777602 speedup is 1.00
Vendor-OS 'suse-11.0-mysystem' has been cloned succesfully.
Vendor-OS 'suse-11.0-mysystem' has been added to DB (ID=3).
```

Das Subkommando `list-installed` zeigt Ihnen alle vorhandenen Stage1-Systeme an:

```
x60s:~ # slxos-setup list-installed
List of installed vendor-OSes:
    suse-11.0-mysystem
    suse-10.3
```

Plugins - die modularen Erweiterungen des Grundpakets

Die generelle Software-Architektur von OpenSLX zielt darauf, sämtliche über die Grundfunktionalität (die Bereitstellung und Verknüpfung der einzelnen Stages) hinausgehende Fähigkeiten des Systems mittels spezieller Erweiterungen (OS-Plugins) zu implementieren. So genügen die bisherigen Schritte für ein bootfähiges Grundsystem, jedoch nicht für einen komfortablen grafischen Desktop. Diese Aufgabe übernehmen die beiden Plugins `xserver` und `desktop`. Das erste sorgt für die automatische Einrichtung von Xorg und die Einbindung der vielleicht gewünschten proprietären OpenGL-Treiber von Nvidia oder ATI/AMD. Das zweite kümmert sich um das Setup des gewünschten Displaymanagers und der grafischen Default-Sitzung.

Die Installation der Plugins und deren Konfiguration, wenn Sie von den Defaults abweichende Einstellungen vornehmen möchten, realisiert das Kommando `slxos-plugin`:

```
x60s:~ # slxos-plugin install suse-11.0 desktop manager=kdm
Plugin desktop has been installed into vendor-OS 'suse-11.0'.
x60s:~ # slxos-plugin install ubuntu-8.04-clone xserver
Plugin xserver has been installed into vendor-OS 'suse-11.0'.
```

Für eine spätere Nachinstallation von Plugins gilt: Ohne einen anschliessenden Export und Aufruf des OpenSLX Demuxers, welche im folgenden erklärt werden, stehen die ausgewählten Plugins auf den bootenden SLX-Clients nicht zur Verfügung.

Stage3-Attribute, die wie beim `desktop` Plugin Einstellungen für den später laufenden Client beinhalten, schreibt `slxos-plugin` in die Datenbank. Dort können Sie diese jederzeit für einzelne Clients, ganze Gruppen oder Vendor-OS anpassen.

Systeme exportieren

Die bisher vorgenommenen Handlungen dienen noch nicht dazu, bootenden Clients ein Rootfilessystem bereitzustellen. Dieses geschieht im nächsten Schritt mit `slxos-export`, welches den Übergang von Stage1-Installationen in Stage2-Exports realisiert. Der Parameter `list-installed` sollte, mit `slxos-export` aufgerufen, das identische Ergebnis wie bei `slxos-setup` liefern:

```
x60s:~ # slxos-export list-installed
List of installed vendor-OSes:
    suse-10.3-mysystem
    suse-11.0
```

Von diesen Systemen können Sie nun einen Export erzeugen, unterstützt sind derzeit die Typen `nfs` und `nbd-squash`. Weitere können im Laufe zukünftiger Entwicklungen hinzukommen. Mit dem Unterkommando `export` legen Sie einen neuen Export an, beispielsweise:

```
x60s:~ # slxos-export export suse-11.0-x60 sqfs-nbd
invoking mksquashfs...
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

```
Parallel mksquashfs: Using 1 processor
Creating little endian 3.0 filesystem on /data/openslx/export/sqfs/suse-11.0-x60
  block size 65536.
[=====] 294751/294751 100%
[ ... ]
Filesystem size 2325395.23 Kbytes (2270.89 Mbytes)
  41.13% of uncompressed filesystem size (5654449.42 Kbytes)
Inode table size 2888177 bytes (2820.49 Kbytes)
  30.34% of uncompressed inode table size (9518178 bytes)
Directory table size 2735640 bytes (2671.52 Kbytes)
  47.98% of uncompressed directory table size (5701677 bytes)
[ ... ]
vendor-OS '/data/openslx/stage1/suse-11.0-x60' successfully exported to
'/data/openslx/export/sqfs/suse-11.0-x60'!
Export 'suse-11.0-x60:sqfs-nbd' has been added to DB (ID=1)...
#####
Please make sure you start a corresponding nbd-server:
      nbd-server 5000 /data/openslx/export/sqfs/suse-11.0-x60 -r
#####
system 'suse-11.0-x60:sqfs-nbd' has been successfully added to DB (ID=1)
```

Ein klassisches NFS-Rootfilesystem legen Sie durch einen ähnlichen Aufruf an:

```
x60s:~ # slxos-export export suse-11.0 nfs
building file list ...
...
... rsync Ausgaben ...
...
sent 5642940230 bytes received 5681918 bytes 9147566.23 bytes/sec
total size is 5624419842 speedup is 1.00
vendor-OS '/data/openslx/stage1/suse-11.0-x60' successfully exported to
'/data/export/nfs/suse-11.0'!
Export 'suse-11.0-nfs' has been added to DB (ID=1)...
#####
Please make sure the following line is contained in /etc/exports
in order to activate the NFS-export of this vendor-OS:
      /data/export/nfs/suse-11.0 *(ro,no_root_squash,async,no_subtree_check)
#####
system 'suse-11.0-nfs' has been successfully added to DB (ID=1)
```

Anschließend können Sie sich durch `slxos-export list-exported` vergewissern, dass alles bis hierhin funktioniert hat.

Quickstart

Bis zu diesem Punkt haben Sie das Rootfilesystem (entweder das klassische NFS oder NBD mit SquashFS) vorbereitet und die Datenbank mit einigen Basisdaten gefüllt. Nun fehlen noch die Bereitstellung der jeweiligen Kernels mit dem dazu passenden InitialRamFS und eine geeignete Client-Konfiguration. Diese Aufgabe übernimmt das Kommando `slxconfig-demuxer`, welches keiner weiteren Kommandozeilenparameter bedarf.

Es arbeitet direkt mit den Einträgen der Datenbank. Mit den Basisdaten erreichen Sie bereits eine gültige Konfiguration, die für jeden anfragenden Client gilt. Für feinere, granularere Einstellungen dient das Werkzeug `slxconfig`, welches im übernächsten Abschnitt besprochen wird.

Ihr ISC-DHCP-Server sollte laufen und die beiden folgenden Einträge für den Netzbereich haben, den Sie stateless booten wollen:

```
next-server 10.8.4.254;
filename "pxe/pxelinux.0";
```

Der zweite Eintrag hängt vom festgelegten TFTP-Rootverzeichnis ab. Wie gehts weiter. Wenn DHCP läuft die entsprechenden Einträge machen. Damit sollten Sie alles vorbereitet haben, um mit den ersten Clients testen zu können. Wenn Sie noch nicht die PXE-Bootfähigkeit eingeschaltet haben, sollten Sie das nun auf Ihren Clients nachholen. Diese sollten nun beim Start eine IP-Adresse zugewiesen bekommen und anschliessend den Kernel und das InitRamFS herunterladen.

Die Datenbank und das Boot-Setup

Sobald Sie mehr Systeme verwalten wollen oder Clients unterschiedliche Einstellungen benutzen sollen, kommt die Konfiguration per Datenbank wieder ins Spiel. Mit der Datenbank, die von `slxos-*` unterhalb von `/var/opt/openslx/db` oder in dem von Ihnen definierten Verzeichnis und Typ angelegt wurde, unterhalten Sie sich mit dem Kommando `slxconfig`. Es kann vorkommen, dass nicht jeder Typ von Datenbank auf jedem Server unterstützt ist. Das stellen Sie bereits am Ende des Aufrufs von `slxos-setup` fest. Dann können Sie mit `slxsettings` das ändern.

In zukünftigen Versionen soll dieses um ein Webfrontend erweitert werden. Die bisherigen Einträge in die Datenbank erfolgten automatisch und zeigen

```
x60s:~ # slxconfig list-vendoros
List of the matching vendor-OSes:
    suse-10.3-mysystem
    suse-11.0
x60s:~ # slxconfig list-export
List of the matching exports:
    suse-10.3-mysystem      (nfs)
    suse-11.0:sqfs-nbd      (sqfs-nbd)
```

für die installierten Vendor-OS, genannt SLX-Stage1 und die daraus erzeugten Exporte. Üblicherweise wird automatisch ein Eintrag für ein System in der Datenbank erzeugt, die:

```
x60s:~ # slxconfig list-systems
List of systems:
    <<<default>>>
    suse-10.3-mysystem:nfs
    suse-11.0:sqfs-nbd
```

anzeigt. Jetzt ...

```
x60s:~ # slxconfig add-client slx-test01 mac=00:01:02:03:04:06
client 'slx-test01' has been successfully added to DB (ID=1)
x60s:~ # slxconfig add-client slx-test02 mac=00:11:22:33:44:55
client 'slx-test02' has been successfully added to DB (ID=2)
```

Damit sich Ihr Administrator auch auf den Clients anmelden kann, sollten Sie ein gemeinsames Root-Passwort festlegen. Für deutschsprachige Nutzer macht es Sinn, die Standardsprache zu ändern:

```
x60s:~ # slxconfig change-system suse-11.0:sqfs-nbd country=de
```

Wenn alles funktioniert, können Sie dieses auch wieder entfernen. Eine Alternative bestünde darin per ConfTGZ einen "Authorized"-SSH-Key auf die Clients verteilen zu lassen, um von einer festgelegten Maschine problemlos zugreifen zu können. Ohne Zugriff auf den Client selbst, ist jedoch das Debugging oft nicht möglich.

Den letzten Schritt vollziehen Sie durch den Aufruf von `slxconfig-demuxer`. Dieses Kommando benötigt keine weiteren Subkommandos oder Parameter, da es alle Informationen direkt aus der Datenbank bezieht. Alle wesentlichen Operationen zum Abschluss der Stage2-Phase werden in diesem Schritt erledigt.

Der `slxconfig-demuxer` baut dabei die Verzeichnisse `<public-path>/tftpboot` und `<public-path>/tftpboot/client-config` bei jedem Durchlauf komplett neu auf. Änderungen, die Sie evtl. in diesem Bereich vorgenommen haben, gehen verloren. Die Quellen des Demuxers sind vielfältig:

- Kernel, -module stammen aus dem Stage1, dem jeweiligen Vendor-OS von dem ein Export erzeugt wurde
- Die Basiskonfiguration, die `initramfs-setup`, für das erstellte InitRamFS wird komplett aus der Datenbank generiert. Einträge ändern Sie mit dem gleich vorgestellten `slxconfig`.
- Das Paket der Konfigurationsdaten (ConfTGZ) stammt aus `/var/opt/openslx/config/<system-name>`. Der Systemname wird beim Eintrag in die Datenbank durch `slxos-export` automatisch generiert. (Client-Konfiguration)

Unterhaltung mit der Datenbank

Das Werkzeug `slxconfig` stellt die Benutzerschnittstelle zur Datenbank dar. Damit können Sie sich alle Einstellungen anzeigen lassen und Änderungen vornehmen. So zeigt Ihnen `slxconfig list-system` alle aktivierten, für Clients potenziell bereitgestellten SLX-Systeme.

Im letzten Abschnitt hatten Sie einzelne Clients in die Datenbank eingetragen. Alle gesetzten Variablen sehen Sie sich mit:

```
x60s:~ # slxconfig --verbose list-system name="<system>"
List of systems:
system 'suse-11.0-main:nfs':
  clients      = <<<default>>>
  comment      = -
  description   = -
  export_id    = 6 (suse-11.0-main:nfs)
  hidden       = -
  id           = 5
  kernel       = vmlinuz
  kernel_params = aufs vga=0x317 quiet
  label        = suse-11.0-main:nfs
PLUGINS:
  30...Theme
  50...VMware
ATTRIBUTES:
  country      = de
  ramfs_fsmods = aufs
  ramfs_nicmods = forcedeth e1000 e100 tg3 r8169 pcnet32 b44 8139too
  slxgrp       = default
  start_cron   = yes
  start_dreshal = yes
  start_nfsv4  = yes
  start_ntp    = yes
  start_sshd   = yes
  start_x      = yes
  start_xdmcp  = kdm
  theme::name  = openslx
  vmware      = nfs://10.4.6.1/vol/vmwareImages/vmware
```

Die meisten Einstellungen sind dabei nicht belegt und werden für die Erstellung der `machine-setup` dann vom Default-System "`<>`" übernommen. Sie können Parameter sowohl Ihres Systems als auch des Defaultsystems ändern:

```
x60s:~ # slxconfig change-system "<<<default>>>" start_xdmcp=gdm
system '<<<default>>>' has been successfully changed.
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

Diese Konfiguration sorgt dafür, dass für das auf einer Klonbasis erstellte SUSE-11.0 alle Clients mit dem Default-Splash in der Auflösung 1024x768 starten:

```
x60s:~ # slxconfig change-system suse-10.3-mysystem:sqfs-nbd \  
kernel_params="vga=0x317 quiet"
```

Wenn das Homeverzeichnis der User auf dem nicht sehr sicheren NFSv3-Weg bereitgestellt werden soll, regelt das der folgende Datenbankeintrag: (*vermutlich nun im Plugin*)

```
x60s:~ # slxconfig change-system suse-10.3-mysystem:sqfs-nbd automnt_dir=/home  
automnt_src="nfs://11.00.4.2/home"
```

In einigen Szenarien verwenden Sie verschiedene Server für DHCP, TFTP und den Fileserver für das Client-Rootfilesystem. Sollte ihr Exportserver vom DHCP-Server abweichen, teilen Sie das der Datenbank mit:

```
x60s:~ # slxconfig change-export suse-10.3-main::sqfs-nbd server_ip=10.30.4.4 port=5002  
export 'suse-10.3-main::sqfs-nbd' has been successfully changed  
x60s:~ # slxconfig list-export suse-10.3-main::sqfs-nbd  
List of exports:  
export 'suse-10.3-main::sqfs-nbd':  
  comment      = -  
  id           = 10  
  port        = 5002  
  server_ip    = 10.30.4.4  
  type        = sqfs-nbd  
  uri         = -  
  vendor_os_id = 7 (suse-10.3-main)
```

```
x60s:~ # slxconfig change-system suse-11.0-clone::nfs scratch="nfs://10.8.4.2/data/export/tmp"
```

System-Update und Ergänzungen

In direkt aus den Paketquellen installierten Stage1-Installationen können Sie Updates auf dem Server vornehmen lassen. Dieses triggert der update-Parameter von slxos-setup:

```
x60s:~ # slxos-setup update suse-11.0-kde  
Setting up Update Process  
Setting up repositories  
base_non-oss      100% |=====| 951 B    00:00  
base              100% |=====| 951 B    00:00  
base_update      100% |=====| 1.2 kB   00:00  
Reading repository metadata in from local files  
primary.xml.gz   100% |=====| 548 kB   00:00  
##### 1340/1340  
Resolving Dependencies  
[ ... etliche weitere Ausgaben ... ]  
Starting SUSEconfig, the SUSE Configuration Tool...  
Running in full featured mode.  
Reading /etc/sysconfig and updating the system...  
Executing /sbin/conf.d/SuSEconfig.perl...  
Executing /sbin/conf.d/SuSEconfig.permissions...  
Finished.  
Vendor-OS 'suse-11.0-kde' updated succesfully.
```

Das Update eines Klon-Systems nehmen Sie auf der Referenzmaschine, im sogenannten Stage0 vor. Anschliessend rufen Sie erneut `slxos-setup clone 11.00.4.50:/ suse-10.3-mysystem auf:`

```
Vendor-OS 'suse-10.3-mysystem' has been re-cloned succesfully. No need to
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

```
change vendor-OS 'suse-10.3-mysystem' in OpenSLX-database.
```

```
x60s:~ # slxos-setup clone 0.20.4.50:/ suse-10.3-mysystem
building file list ...
[ ... etliche weitere Ausgaben ... ]
sent 434920 bytes received 121491994 bytes 375737.79 bytes/sec
total size is 8644862874 speedup is 70.90
Vendor-OS 'suse-10.3-mysystem' has been cloned succesfully.
No need to change vendor-OS 'suse-10.3-mysystem' in OpenSLX-database.
```

Als Abschlussmeldung lesen Sie dann, dass `slxos-setup` den Vorgang erfolgreich abschliessen konnte. Für System-Klone gibt es keinen speziellen Update-Parameter, da das Update ausserhalb des Servers auf der Referenzmaschine erfolgt und erst dann auf den Server kopiert wird.

Aufräumarbeiten

Ältere Systeme oder Testversionen wollen Sie vielleicht irgendwann wieder loswerden. Dazu haben die beteiligten Tools `slxos-setup` und `slxos-export` eine Entfernen-Option. Sie gehen dabei in umgekehrter Reihenfolge vor. `slxconfig` behandelt die Datenbank und nimmt sonst keine Löschungen im Dateisystem vor.

```
x60s:~ # slxconfig remove suse-11.0
removing vendor-OS folder '/data/openslx/stage1/suse-11.0'...
Vendor-OS 'suse-11.0' removed succesfully.
Vendor-OS 'suse-11.0' has been removed from DB!
```

Die Entfernung von Systemen aus dem `~/tftpboot` Bereich erfolgt beim erneuten Aufruf von `slxconfig-demuxer`.

Im nächsten Schritt entfernen Sie die Exporte (Teile des Stage2), welche üblicherweise nicht unerheblich Speicherplatz beanspruchen. Dieser Vorgang löscht gleichzeitig den zugehörigen Datenbank-Eintrag. (*ändern!*)

```
x60s:~ # slxos-export remove suse-11.0 nbd
export '/srv/openslx/export/nbd-squash/suse-11.0' successfully removed!
Export 'suse-11.0' has been removed from DB.

x60s:~ # slxos-export remove suse-11.0
export '/srv/openslx/export/nfs/suse-11.0' successfully removed!
Export 'suse-11.0' has been removed from DB.
```

Im letzten Schritt werfen Sie das installierte Basis-System (Stage1) von der Platte ihres Servers. Dieses folgt analog und entfernt ebenso den Datenbankeintrag.

```
x60s:~ # slxos-setup remove suse-11.0
Vendor-OS 'suse-11.0' removed succesfully.
Vendor-OS 'suse-11.0' has been removed from DB!
```

Bootablauf

PXE-Boot

Was lange Zeit "richtigen" Unix-Workstations vorbehalten war, können nun auch Ihre PCs: Intels Pre boot eXtension Environment (PXE) stellt eine Variante dar, neben dem Start von Platte oder DVD über das Netz eine festplattenlose Maschine zu booten. Mittels DHCP beschafft sich ein netzbootender Rechner eine IP-Adresse und die Information, wo er per TFTP ein PXE-Image laden kann. Dieses stellt dann seinerseits weitere Bootfunktionalität zur Verfügung. An diesem Punkt setzt eine ganze Klasse von Netzwerk-Software

an. Recht bekannt sind inzwischen die vielfältigen Möglichkeiten den netzbasierten Betriebssystem-Installation ob Linux oder Windows.

Die PXE-Bootmenüs für Ihre Clients richtet der `slxconfig-demuxer` vollautomatisch im Verzeichnis `~/tftpboot/pxe/pxelinux.cfg` ein. Wenn Sie das Aussehen Ihrer Menüs oder das Default-Passwort für den Zugriff auf die jeweilige Kernel Commandline anpassen wollen, können Sie dieses in durch Anlegen bzw. Verändern der Datei `/etc/opt/openslx/PXE-template` erreichen.

DHCP für die IP-Adressverwaltung

Wenn es in Ihrem Netz noch keinen DHCP-Server gab, ist nun der richtige Moment gekommen, einen aufzusetzen. In den meisten Fällen genügt es für den Start des DHCP-Servers zwei Dateien anzupassen. Die Eintragungen für Ihre Clients finden Sie in der `/etc/dhcpd.conf`. Für erste Tests genügen die folgenden Eintragungen:

```
ddns-update-style none;
subnet 10.8.4.0 netmask 255.255.255.0 {
    server-identifier 10.8.4.1;
    next-server 10.8.4.1;
    filename "pxe/pxelinux.0";
    option routers 10.8.4.254;
    range 10.8.4.101 10.8.4.200;
}
```

Anschliessend legen Sie in der `/etc/sysconfig/dhcpd` fest, dass der DHCP-Server auf dem richtigen Interface, im Beispiel `eth0`, lauscht und zur Sicherheit in einer `chroot` Umgebung läuft:

```
DHCPD_INTERFACE="eth0"
DHCPD_RUN_CHROOTED="yes"
```

TFTP

Der Startvorgang eines Clients bis zu und in Stage3 ist stark mit TFTP verknüpft. Die Initialdaten (PXE-Linux und das Menüsystem) sowie die Client-Konfiguration werden mittels dieses Protokolls geholt. In einer typischen SLX-Installation sieht das Verzeichnis für TFTP so aus:

```
tftpboot/client-config
tftpboot/client-config/suse-11.0-A
tftpboot/client-config/suse-11.0-A/01-<MAC>.tgz
tftpboot/client-config/suse-11.0-A/default.tgz
tftpboot/client-config/<system>/...
tftpboot/pxe
tftpboot/pxe/menu.c32
tftpboot/pxe/pxelinux.0
tftpboot/pxe/suse-11.0
tftpboot/pxe/suse-11.0/initramfs-1
tftpboot/pxe/suse-11.0/vmlinuz
tftpboot/pxe/<vendor-os>/...
tftpboot/pxe/...
```

Das Verzeichnis liegt per Default unterhalb `/srv/openslx`, kann aber in seiner Position im Dateisystem durch `slxsettings --tftpboot-path=Neuer-Pfad` geändert werden.

Bei den meisten Distributionen ist als TFTP-Server der `in.tftp` dabei. Diesen aktivieren Sie üblicherweise mittels `xinetd`. Hierfür editieren Sie je nach Distribution die Datei: `/etc/xinetd.d/tftp`

```
service tftp
{
```

OpenSLX - Schritt-für-Schritt-Anleitung - OpenSLX Lab

```
socket_type      = dgram
protocol         = udp
wait             = yes
user             = root
server           = /usr/sbin/in.tftpd
server_args      = -s /tftpboot
disable         = yes
}
```

Die vorgenommenen Einstellungen für das Verzeichnis müssen zu den Einstellungen des OpenSLX-Paketes und des DHCP-Servers passen. Die Variable `filename` enthält `/tftpboot/pxe/pxelinux.0`, diese Datei soll der TFTP-Server den anfragenden PXE-Clients liefern.

Der Eintrag `-s /tftpboot` bedeutet, dass beim Start des `in.tftpd` unterhalb des eingetragenen Pfades Clients Dateien vom Server herunterladen dürfen. Damit der Client auch weiss, wo er suchen muss, teilt ihm dieses der DHCP-Server in der Variablen `filename` vorher mit. PXE selbst ist nicht in der Lage sofort einen grossen Kernel von durchaus mehreren Megabyte Umfang zu laden. Deshalb bekommt es einen speziellen Boot-Loader aus dem Syslinux-Paket von Peter Anwin geliefert `pxelinux.0`.

Das Tool übernimmt alle weiteren Schritte und kann dabei noch ein Bootmenü aufbauen und weitere Bootoptionen zur Verfügung stellen. Nach der Einrichtung starten Sie den `xinetd` neu und überprüfen mit dem kleinen Kommando `tftp` von einem beliebigen Rechner in Ihrem Netz oder dem Server direkt, ob Sie die Datei `pxelinux.0` von der Stelle, die Ihre Clients erwarten, laden können:

```
dirk@x60s:/tmp> tftp 10.8.4.254
tftp> get /tftpboot/pxe/pxelinux.0
tftp> quit
dirk@x60s:/tmp> ls -la pxelinux.0
-rw-r--r-- 1 dirk users 13320 2007-04-20 16:10 pxelinux.0
```

Syslinux

Im letzten Abschnitt wurde ein Programm zum Darstellen von Boot-Menüs und Laden von Kernel und InitialRamFS per TFTP angesprochen: `pxelinux.0` ist Bestandteil des größeren Syslinux-Paketes. Das zentrale Programm `pxelinux.0` und Menü-Hilfsprogramme stehen im OpenSLX-Paket vorkompiliert zur Verfügung.

Für PXElinux gibt es eine paar spezifischer Parameter. `pxelinux.0` erwartet seine Konfigurationsdatei im Verhältnis zur eigenen Lage im Unterverzeichnis `pxelinux.cfg`. Dort lädt es, wenn es keine speziellen maschinenspezifischen Konfigurationsdateien findet, eine Datei namens `default`. Maschinenspezifische Dateien haben das Format `01-<MAC-des-Clients>`. Das Format der spezifischen und `default`-Dateien ist identisch, der innere Aufbau darf natürlich variieren. Sie werden auf Ihrem System Einstellungen finden, die fürs erste analog aussehen sollten:

```
NOESCAPE 0
PROMPT 0
TIMEOUT 10
DEFAULT menu.c32
IMPLICIT 1
ALLOWOPTIONS 1
MENU TITLE Was möchten Sie tun (Auswahl mittels Cursortasten)?
MENU MASTER PASSWD secret
LABEL memtest
    KERNEL memtest.bin
```

Einen ersten Test eines PXE-Boots über Netzwerk unternimmt man am besten mit dem kleinen Test-Tool `memtest`. Es ist ein kleines Binärprogramm, welches intensive Speichertests in endlosen Testläufen durchführt. Damit man nicht immer nach einer CD mit der aktuellen Version fahnden muss, kann man es

leicht für alle PXE-fähigen Maschinen im Netz bereitstellen. Das Tool findet sich beispielsweise auf einer der Heft-DVDs oder den ersten SUSE-Installations-CDs und sollte in das Verzeichnis kopiert werden, wo `pxelinux.0` liegt. Soll es an einer anderen Stelle liegen, so muss sichergestellt sein, dass der TFTP-Server auf dieses Verzeichnis zugreifen darf. Dann kann man auch einen absoluten Pfad, wie `kernel/tftpboot/tools/memtest.bin` angeben. Die Endung vom `memtest` kann frei gewählt werden, darf aber nicht auf `"*.0"` lauten. Wenn alles funktioniert hat, kann man im nächsten Schritt mit einem Linux-Kernel und einer dazu passenden Ramdisk experimentieren.

```
prompt 1
timeout 100
default dxs
label dxs
    kernel dxs
    append vga=normal initrd=initrd-dxs nfsserver=10.30.4.1:/nfsroot/ldc
    ipappend 1
label memtest
    kernel memtest.bin
```

Die meisten Parameter sind selbsterklärend: Prompt sorgt dafür, dass PXElinux mit dem Prompt "boot:" für 100 Sekunden wartet (`timeout 100`). Wenn in dieser Zeit keine Eingabe erfolgte, wird als default das "label dxs" gestartet. Die Einstellung geht davon aus, dass im Unterverzeichnis `/nfsroot/ldc/boot` der Kernel mit dem Namen `kernel-dxs` und eine dazu passende Ramdisk mit dem Namen `initrd-dxs` liegen. Das erreicht man am einfachsten wie folgt:

```
cp /boot/vmlinuz /nfsroot/ldc/boot/kernel-dxs
cp /boot/initrd(.img) /nfsroot/ldc/boot/initrd-dxs
```

"`ipappend 1`" sorgt dafür, dass PXE die mittels DHCP ermittelte Konfiguration in die Kernel-Commandline schreibt. Diese enthält dann im Beispiel bleibend den zusätzlichen Parameter "ip":

```
ip=11.00.40.2:11.00.40.71:11.00.40.254:255.255.255.0
```

Den Inhalt dieser speziellen Zeile kann man sich später mit dem Kommando `cat /proc/cmdline` ansehen. Nachdem die Konfiguration, wie eben beschrieben erstellt wurde, sieht man beim Neustart des Test-Clients schon einiges mehr. Nun sieht der Client fast schon wie gewohnt aus: Ein Bootloader kümmert sich um den Kernel sowie die Initial Ramdisk und sorgt für deren Start. Der Kernel entpackt sich, wie von Knoppix oder einer festplattenbasierten Installation gewohnt. Die Mini-Linux-Umgebung der Initial-Ramdisk wird ausgeführt. Jedoch bricht der Prozess mit einem fehlgeschlagenen Mount des Root-Dateisystems ab.

Bild: `vmware-normalinitrd.png`

Das liegt einfach daran, dass Kernel und Ramdisk von einer klassischen Installation stammen und für den Start von Festplatte ausgelegt sind. An dieser Stelle ist der Admin schon ziemlich weit und verfügt über das Verständnis der meisten Grundideen, um nun Diskless Linux Projekte ausprobieren zu können.

Konfiguration von OpenSLX Clients

Anders als bei klassischen Linux-Workstations können OpenSLX Clients ihre Konfiguration nicht lokal ablegen. Deshalb richten sich die Maschinen in mehreren Schritten selbst ein:

1. Sie erhalten ihre Basis-IP-Konfiguration beim Booten per PXE oder Etherboot/gPXE, wenn in der PXE-Datei der Parameter `IPAPPEND 1` gesetzt ist. Andernfalls ruft die Maschine im Stage3 erneut einen DHCP-Client auf.
2. Wenn der Kernel das Stage3-Init gestartet hat, liest das Hauptskript die Konfigurationsdaten des frühen Stadiums aus der Kernel Commandline und der im

InitRamFS mitgelieferten `machine-setup`-Datei.

3. Anschließend stehen für eine erweiterte Konfiguration innerhalb von Stage3 weitere Wege je nach eingestellter Konfigurationsart zur Verfügung.

4. Sodann lesen die Stage3-Skripten die geladene Konfiguration ein und erzeugen daraus für Stage4 die überwiegende Zahl der Einstellungen.

Alle wesentlichen Konfigurationsschritte finden also während des Ablaufs des InitialRamFS statt. Die zentrale Konfigurationsdatei ist `machine-setup`. Sie wird beim Setup des InitialRamFS in Stage2 initial und während des Stage3 weiter gefüllt und ausgewertet. In Stage4 steht sie weiterhin im `/etc` Verzeichnis zur Verfügung.

Die Kernel Commandline

Der `slxconfig-demuxer` richtet neben dem InitialRamFS und der Client-Konfiguration ebenso die Kernel Commandline ein. Sie besteht aus notwendigen und optionalen Einträgen, welche per PXElinux oder Etherboot übergeben werden. Hierzu kann die IP-Konfiguration und muss das InitialRamFS zählen. Jedoch gibt es ein Problem, welches in der Begrenzung der Kernel Commandline bei Intel-Plattformen auf 256 Zeichen liegt, weshalb Sie diese Zeile nicht beliebig lang machen können.

- `debug` - ohne Angabe eines Levels, setzt den Debuglevel auf 1
- `debug=...` - setzt den Debuglevel auf den angegebenen Integerwert. Für eine Auflistung aller Debuglevel siehe Anhang E.2
- `nodhcp` - kein erneutes DHCP zur Konfiguration im InitRamFS; Einstellung für Fälle, in denen die Standard IP und DNS Daten auf anderem Wege bereitgestellt werden (ConfTGZ und PXE-IP-Info). Defaultmäßig gibt es diesen Eintrag nicht in der Kernel Commandline.
- `ldap=...` - verwende LDAP zur Konfiguration (URI Syntax)
- `file` - verwende TFTP zum Beschaffen von Konfigurationsdatei(en)
- `file=...` - verwende TFTP und nimm angegebenen TFTP-Server und Pfad (URI Syntax)
- `ldsc` - generiere keine `ld.so.cache` und übernimmt die vorhandene3 aus Stage2 übernommene.
- `rootfs=...` - Art des zu verwendenden Rootfilesystems des Clients, infrage kommen derzeit NFS, (D)NBD mit SquashFS. Es wird das URI-Format erwartet, beispielsweise `rootfs=nfs://server/pfad` oder `rootfs=nbd://server:port Filesystem@`. Setzt sich das Rootfilesystem aus mehreren Komponenten zusammen, wird "ldsc" automatisch aktiviert. Statt einer fest eingetragenen Server-IP können Sie die Variable "`@@serverip@@"` verwenden, wenn DHCP- und Rootfilesystem-Server identisch sind. Sie wird im Stage3 mit dem via DHCP empfangenen Wert für den DHCP-Server ersetzt.
- `dcsize=...` - nur sinnvoll bei definiertem `rootfs=dnbd://...`, definiert die Größe des Cachefiles für DNBD. Da das Cachefile derzeit im RAM des Clients angelegt wird, sollte es passend zur RAM-Ausstattung gewählt werden.
- `ip=...` - wird durch PXE-Linux bzw. Etherboot/gPXE bestückt.
- `tmpfssize=...` - Größe des max. temporären Speicherplatzes im RAM (tmpfs)
- `vci=...` - Vendor Code Identifier, nur sinnvoll, wenn DHCP benutzt wird

Zentrale Konfigurationsdatei machine-setup

Die zentrale Konfigurationsdatei eines OpenSLX Clients ist `machine-setup`. Sie ist in der aktuellen Version in `/var/opt/openslx/config/default/initramfs/machine-setup` zu finden. Sie sollte für eine gegebene Installation mit sinnvollen Standardeinstellungen vorbelegt werden, da sie als Initial-Konfiguration direkt in das erzeugte InitRamFS eingefügt wird. Die im Anschluss beschriebenen Konfigurationsmethoden hängen an diese Datei ihre Ergebnisse an, so dass beispielsweise die DHCP-Konfiguration die evtl. vorbelegten Variablen für Hostname, DNS-Server, ... überschreiben können. Variablen sind dann unter Umständen mehrfach definiert. Es gilt jedoch immer nur die letzte, so dass die Datei von hinten her zu lesen ist, wenn man eine bestimmte Zuweisung sucht. Statt einer fest eingetragenen Server-IP für die verschiedenen Quellen oder Services kann die Variable "`@serverip@`" in `@machine-setup` verwendet werden. Sie wird im Stage3 mit dem via DHCP empfangenen Wert für die Adresse des DHCP-Servers ersetzt. Damit kann man sich statische Einträge sparen, wenn sowieso für wichtige bzw. alle SLX-Services die identische IP gilt. Obwohl in den meisten Umgebungen OpenSLX Clients sehr einheitlich gehalten werden, kann es doch von Interesse oder notwendig sein, für einzelne Clients separate Festlegungen zu treffen. Das wird insbesondere in größeren Netzwerken mit Gruppen von Clients immer wichtiger. Alle der im Folgenden genannten Konfigurationsarten greifen in Stage3 des Client-Starts.

Derzeit sind drei (vier) Konfigurationswege vorgesehen, wenn auch noch nicht alle implementiert:

- via TFTP (configuration plus extensions - Client-Konfiguration) `file get` - ist wesentlicher Bestandteil der aktuellen Version 4. Ergänzungen zur datenbank-generierten `machine-setup`, wie User-Authentifizierung, Home-Verzeichnisse und admin-spezifische Skripten (zusammengefasst als TGZ) versucht der Client (MAC sei `00:11:43:7c:da:ff`) nacheinander in der Form `01-00-11-43-7c-da-ff.tgz` und `default.tgz` zu beschaffen. Die Dateien werden am Anfang der Services-Konfiguration eingespielt (`servconfig`), können also u.U. noch von dieser modifiziert werden. Dateien können explizit angegeben werden: `file=tftp://<server-ip>/pxe/client-config/<system-name>/01-00-` oder sollten nach einem speziellen Schema präsent sein.

Deshalb sollte man bei auftretenden Problemen zuerst auf dem Client überprüfen, ob die Commandline mit allen Optionen vollständig vorliegt.

```
apt-get install libdbd-sqlite3-perl sqlite3
```

Probleme

```
slxconfig-demuxer --verbose-level=2
```

```
slxos-setup remove suse-11.0-x86_64
removing vendor-OS folder '/var/opt/openslx/stage1/suse-11.0-x86_64'...
Vendor-OS 'suse-11.0-x86_64' removed succesfully.
```

Also available in: [HTML](#) [TXT](#)

Loading...

Powered by [Redmine](#) © 2006-2009 Jean-Philippe Lang