Unbeaufsichtigte Installation eines Linux (Ubuntu) Clients für Schulungsräume mit Hilfe von PXE und preseed-Dateien

Das Vorhaben

Es soll ein Server installiert werden, der unbeaufsichtigt eine Installation eines Linux-Clients (in meinem Fall eine Ubuntu 12.04 LTS-Version (mit einem 3.11.0.17 Kernel) durchführt. Es sind verschiedene Programme zusätzlich zu installieren (z.B. Rapidminer und Corsaro) die zusätzliche Pakete benötigen. Auf den Clients soll ein Datenaustausch nicht möglich sein (also vollkommene Trennung vom Internet, USB-Speicher sollen nicht funktionieren, WLAN aus etc.).

- Die Home-Verzeichnisse der User sollen zentral auf dem Server abgelegt werden und natürlich auch gesichert werden.
- User werden in unregelmässigen Abständen für die Schulungen neu erstellt und sollen ein zufälliges Passwort erhalten. Das soll der Einfachheit halber per Script erfolgen.
- Eine (oder auch mehrere) Linux-Live-DVD's sollen ebenfalls bootbar sein.

Vorweg-Überlegung

Ich sehe insbesondere für die Prüfungszwecke 2 Möglichkeiten (jaja, es gibt natürlich mehrere).

1. Einen Terminal-Server mit „Thin-Clients“
2. Eine „Fat-Client“ Lösung mit zentraler Userverwaltung auf dem Server

Der Terminal-Server hat den Vorteil, dass man relativ schwache Clients verwenden kann, da der Großteil der Rechenleistung auf den Server ausgelagert wird. Außerdem sind alle Software-Pakete zentral auf dem Server und Updates etc. müssen nur dort gemacht werden. Weiters brauchen Thin-Clients keine Festplatte, da alles am Server liegt

Die Fat-Client Lösung hat den Vorteil, dass der Server relativ „schwach“ sein darf (Software wird auf dem Client gestartet) und sich die CPU-Last auf dem Server sehr in Grenzen hält. Das Gerät funktioniert prinzipiell auch dann, wenn der Server nicht verfügbar ist.

Tatsächlich habe ich auch mit einem LTSP (Linux Terminal Server Project) begonnen, bin aber später aus mehreren Gründen zu den Fat-Clients über gegangen.

Ich gehe *nicht* darauf ein, wie ein LTSP installiert wird. Das Debian LTSP Paket hat mir aber einen isc-DHCP-Server und einen TFTP-Server installiert, den ich gleich für mein „Fat-Client“-Projekt genutzt habe. Ausserdem habe ich so einen Terminal-Server als Notlösung. Als DHCP kann natürlich jeder Andere Server auch dienen.

Die Basis-Installation eines LTSP-Servers ist denkbar einfach und wird von mir daher empfohlen.

Ausgangsbasis (was brauchen wir)

- Natürlich einen Server (mit 2 Netzwerkkarten).
- Ich nutze eine 4TB große „Langlauf“-Server-Platte für aktuelle home-Verzeichnisse und gleichzeitig als tägliches „Snapshot“ Backup. (Das Backup ist nicht als Sicherung im Sinne einer Datensicherung bei Plattendefekt gedacht, viel mehr wenn ein User etwas löscht, sollen die Daten wieder herstellbar sein). Ein echtes Backup ist separat zu lösen.
- Ein relativ aktuelles Debian.
- DHCP-Server (isc-dhcp-server)
- TFTP-Server
- NIS (Zentrale Verwaltung der User und Passwörter)
- Syslinux (PXE-Boot)
- nfs-kernel-server (home Verzeichnisse der User einbinden)
- Apache Webserver (stellt die preseed-Dateien und Installationspakete für weitere Software zur Verfügung)
- IP-Tables
- Internetanbindung
- Switch für die Clients
- Natürlich viele PXE bootbare Clients

A. Grundkonfiguration Server

DHCP-Server

Ich nutze den isc-dhcp-server

```
# apt-get -y install isc-dhcp-server
```

Der Server verfügt über 2 Netzwerkkarten

eth0 = öffentliches Netz (ab hier a.b.c.d)

eth1 = 192.168.67.1

Der Server läuft unter der Domain ltsp.mydomain.tld

Meine ***/etc/network/interfaces*** sieht also so aus:

```
auto eth0
iface eth0 inet static
address a.b.c.d
netmask 255.255.255.0
network a.b.c.0
broadcast a.b.c.255
gateway a.b.c.1
dns-nameservers
dns-search domain.tld

auto eth1
iface eth1 inet static
address 192.168.67.1
netmask 255.255.255.0
```

Damit alles auch über den Namen erkannt werden kann:

/etc/hosts

```
127.0.0.1   localhost
127.0.1.1   ltsp.mydomain.tld   ltsp
a.b.c.d     ltsp.mydomain.tld   ltsp
```

/etc/dhcp/dhcpd.conf

```
ddns-update-style none;

option domain-name "domain.tld";
option domain-name-servers 192.168.67.1,other domain server;
option nis-domain "ltsp.mydomain.tld";

default-lease-time 60000;
max-lease-time 720000;

authoritative;

# allow booting;
# allow bootp;

subnet 192.168.67.0 netmask 255.255.255.0 {

    interface eth1;
    range 192.168.67.66 192.168.67.127;
    option domain-name-servers 192.168.67.1,other domain server;
    option nis-domain "ltsp.mydomain.tld";
    option routers 192.168.67.1;
    option nis-servers 192.168.67.1;
    option broadcast-address 192.168.67.255;
    filename "/pxelinux.0";
    option subnet-mask 255.255.255.0;
    next-server 192.168.67.1;
}
```

other domain server (Zeile 4) kann durch weitere Server ersetzt werden.

NFS-Server

```
# apt-get -y install portmap nfs-common nfs-kernel-server
```

Nun ein install-Verzeichnis anlegen, in dem unsere bootbaren CD's/DVD's liegen werden.

```
mkdir /srv/install
```

Das install-Verzeichnis und das home-Verzeichnis freigeben.

/etc/exports

```
/srv/install 192.168.67.0/24(ro,async,no_root_squash,no_subtree_check)
/home  192.168.67.0/24(rw,async,no_root_squash,no_subtree_check)root@ltsp:/etc#
```

Eine Besonderheit bilden meine Userverzeichnisse in ***/home***, die auf einer 4TB-Platte ausgelagert sind die auch gleichzeitig eine Art täglichen Snapshot als Backup macht.

Die 4 TB Platte (bei mir `UUID=6b5b1050-6d93-4475-b1eb-c823a2edcaa1`) wird in das Verzeichnis ***/backup*** gemountet. Unter **/backup** habe ich ein Verzeichnis ***/home*** erstellt (also **/backup/home**), dass ich noch einmal zusätzlich als ***/home*** einbinde

/etc/fstab

```
#  Backup
UUID=6b5b1050-6d93-4475-b1eb-c823a2edcaa1 /backup    ext4   defaults      0      2
#/home einbinden
/backup/home  /home  none   bind   0      0
```

Es ist an der Zeit den Rechner (b.z.w die Dienste DHCP und NFS) neu zu starten.

```
# /etc/init.d/isc-dhcp-server restart
# /etc/init.d/nfs-kernel-server restart
```

TFTP Server

```
# apt-get -y install tftpd-hpa
```

/etc/default/tftpd-hpa

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

Ich verlege das Standardverzeichnis für den TFTP-Server nach ***/srv/tftp***. Es ist also das Verzeichnis ***/srv/tftp*** zu erstellen (*mkdir /srv/tftp*).

Syslinux

```
# apt-get -y install syslinux
```

Damit unsere Clients ein System vorfinden, wenn sie per PXE booten, muss das Systemfile pxelinux.0 zur Verfügung gestellt werden. Dieses Minisystem wird geladen und soll dann ein Menü zur Verfügung stellen, aus dem wir unsere gewünschte Installation auswählen können.

```
# cp /usr/lib/syslinux/pxelinux.0 /srv/tftp
# cp /usr/lib/syslinux/menu.c32 /srv/tftp
# cp /usr/lib/syslinux/vesamenu.c32 /srv/tftp
```

PXE-Linux sucht nach einem Verzeichnis ***/srv/tftp/pxelinux.cfg*** und darin eine Datei **default**, die unser erstes Menü enthält. Sollte diese Datei (oder auch das Verzeichnis) nicht vorhanden sein, muss man sie anlegen. Diese Datei bildet im Wesentlichen unser Bootmenü ab und stellt uns eine Auswahl an Bootoptionen zur Verfügung.

```
# touch /srv/tftp/pxelinux.cfg/default
```

Der Inhalt dieser Datei sieht wie folgt aus:

```
DEFAULT menu.c32
TIMEOUT 600
ONTIMEOUT BootLocal
PROMPT 0
MENU INCLUDE pxelinux.cfg/pxe.conf
NOESCAPE 1
LABEL BootLocal
        localboot 0
        TEXT HELP
        Boot to local hard disk
        ENDTEXT
MENU BEGIN Ubuntu
MENU TITLE Ubuntu LTS
        LABEL Previous
        MENU LABEL Previous Menu
        TEXT HELP
        Return to previous menu
        ENDTEXT
        MENU EXIT
        MENU SEPARATOR
        MENU INCLUDE Ubuntu/Ubuntu.menu
MENU END
```

Menü1

Dieses Menü ist unser Startmenü, dass ein booten von der lokalen Platte oder ein Untermenü mit diversen Ubuntu Optionen zur Auswahl stellt.

```
MENU INCLUDE ubuntu_lts/Ubuntu.menu
```

bindet ein Untermenü ein (siehe Kasten Menü2 etwas weiter unten), in dem die verschiedenen Bootparameter für unsere Installationen der Clients angelegt sind, dazu kommen wir jetzt.

Bootmedien und Netboot anlegen

```
# cd /srv/tftp
# mkdir Ubuntu
# mkdir ubuntu12_04
```

Ubuntu ist ein Verzeichnis in dem etwas später eine komplette DVD liegen soll. Diese DVD entspricht einer Live-DVD und kann am Client mit allen Optionen gestartet werden als ob sie direkt von DVD gestartet wurde.

ubuntu12_04 enthällt ein minimales Netboot-Image (ähnlich einer Netzwerk Installations-CD) und dient unserer eigentlichen unbeaufsichtigten Netzwerkinstallation.

Wir laden uns von http://cdimage.ubuntu.com/netboot/12.04/ unser gewünschtes Netboot-Image herunter (in meinem Fall das amd64 Paket. Wer einen passenden Kernel booten möchte kann aus mehreren Paketen mit passendem Kernel auswählen. Ich habe eines der netboot.tar.gz-File geladen und entpackt.

Nach dem entpacken des tar.gz-Files wird der gesamte Inhalt des Ordners netboot in das Verzeichnis ***/srv/ubuntu12_04*** kopieren.

> *Anmerkung:*
> *Ich definiere etwas weiter unten (unter preseed-Datei) einen eigenen Kernel direkt in der preseed-Datei. Es kann also praktisch jedes netboot-Image gezogen werden....*

Im Verzeichnis ***/srv/tftp/Ubuntu*** legen wir nun eine Datei mit dem Namen **Ubuntu.menu** (Achtung auf Grossbuchstaben) mit folgendem Inhalt an:

```
LABEL 5
        MENU LABEL Boot Ubuntu 12.04 LTS (64-bit) DVD
        KERNEL Ubuntu/12.04/amd64/vmlinuz.efi
        # nfs freigabe der DVD Dateien
        APPEND boot=casper netboot=nfs nfsroot=192.168.67.1:/srv/install/Ubuntu/12.04/amd64 initrd=Ubuntu/12.04/amd64/initrd.lz
        TEXT HELP
        Boot the Ubuntu 12.04 LTS DVD
        ENDTEXT
LABEL 4
        MENU LABEL Install Ubuntu Client 12.04 LTS (64-bit) English
        # FTP-Freigaben
        KERNEL ubuntu12_04/ubuntu-installer/amd64/linux
      append auto=true netcfg/choose_interface=em1 priority=critical locale=en_US.UTF-8 debian-installer/country=US
debian-installer/language=en debian-installer/keymap=de-latin1-nodeadkeys console-keymaps-at/keymap=de-latin1-nodeadkeys preseed/url=http://192.168.67.1/u12_04_en.cfg
ramdisk_size=14984 initrd=ubuntu12_04/ubuntu-installer/amd64/initrd.gz vga=normal
        TEXT HELP
        Kernel 3.11.0.17
        ENDTEXT
LABEL 3
        MENU LABEL Install Ubuntu Client 12.04 LTS (64-bit) German
        KERNEL ubuntu12_04/ubuntu-installer/amd64/linux
      append auto=true netcfg/choose_interface=em1 priority=critical locale=de_DE debian-installer/country=DE
debian-installer/language=de debian-installer/keymap=de-latin1-nodeadkeys console-keymaps-at/keymap=de-latin1-nodeadkeys preseed/url=http://192.168.67.1/u12_04.cfg
ramdisk_size=14984 initrd=ubuntu12_04/ubuntu-installer/amd64/initrd.gz vga=normal
        TEXT HELP
        Kernel 3.11.0.17
        ENDTEXT
LABEL 2
        MENU LABEL Install Ubuntu Dektop (64-bit) English
        KERNEL ubuntu12_04/ubuntu-installer/amd64/linux
      append auto=true netcfg/choose_interface=em1 priority=critical locale=en_US.UTF-8 debian-installer/country=US
debian-installer/language=en debian-installer/keymap=de-latin1-nodeadkeys console-keymaps-at/keymap=de-latin1-nodeadkeys preseed/url=http://192.168.67.1/u12_04_en_desk.cfg
ramdisk_size=14984 initrd=ubuntu12_04/ubuntu-installer/amd64/initrd.gz vga=normal
        TEXT HELP
        Kernel 3.11.0.17
        ENDTEXT
LABEL 1
        MENU LABEL Install Ubuntu Dektop (64-bit) German
        KERNEL ubuntu12_04/ubuntu-installer/amd64/linux
      append auto=true netcfg/choose_interface=em1 priority=critical locale=de_DE debian-installer/country=DE
debian-installer/language=de debian-installer/keymap=de-latin1-nodeadkeys console-keymaps-at/keymap=de-latin1-nodeadkeys preseed/url=http://192.168.67.1/u12_04_desk.cfg
ramdisk_size=14984 initrd=ubuntu12_04/ubuntu-installer/amd64/initrd.gz vga=normal
        TEXT HELP
        Kernel 3.11.0.17
        ENDTEXT
```

Menü2

Die Label 4 bis 1 unterscheiden sich im Wesentlichen nur in 2 Punkten, es wird für jede Installation eine eigene preseed Datei geladen und je nach Sprache ein anderer Wert für locale gesetzt.

Obiges Untermenü erlaubt es, je nach Wunsch, eine Englisch oder Deutschsprachige Installation durchzuführen. Ausserdem gibt es die Möglichkeit zwischen einem Client (also jenen mit den home-Verzeichnissen am Server) oder einer normalen Desktop-Installation zu wählen.

Zusätzlich (der oberste Menüpunkt) kann die schon erwähnte DVD gestartet werden.

Preseed-Datei (Apache)

Damit unsere Installation auch wirklich vollautomatisch abläuft, müssen alle Angaben, die wir normalerweise händisch machen (Username/Passwort, Festplattenaufteilung etc.) an die Installation übergeben werden. Debian/Ubuntu macht das mit sogenannten preseed-Dateien. Diese Dateien werden per http an den Client weiter gereicht.

```
preseed/url=http://192.168.67.1/u12_04_...cfg
```

```
# apt-get -y install apache2
# cd /var/www
# touch u12_04.cfg
```

Die Datei u12_04.cgf kann mit einem beliebigen Editor geöffnet und bearbeitet werden. Die Datei für die deutschsprachige Installation sieht wie folgt aus:

Genauere Infos: https://help.ubuntu.com/10.04/installation-guide/i386/preseed-contents.html

```
# locale
# eine deutsche Version installieren
d-i debian-installer/locale string de_DE.UTF-8
d-i debian-installer/keymap select de-latin1
#
# fuer eine englische Version installieren
#d-i debian-installer/locale string en_US.UTF-8
#d-i languagechooser/language-name-fb select american

d-i console-keymaps-at/keymap select de
d-i languagechooser/language-name-fb select German
d-i countrychooser/country-name select Germany
d-i keyboard-configuration/layoutcode string de

d-i clock-setup/utc boolean true
d-i time/zone string Europe/Vienna
d-i clock-setup/ntp boolean true
d-i console-setup/ask_detect boolean false

## Interface
d-i netcfg/choose_interface select auto
# langsamer dhcp-server ??
d-i netcfg/dhcp_timeout string 60

# den wireless adapter weg !!!
ethdetect ethdetect/use_firewire_ethernet boolean false
d-i netcfg/wireless_wep string

# firmware laden, wenn noetig !!
d-i hw-detect/load_firmware boolean true
```

```
# Any hostname and domain names assigned from dhcp take precedence over
# values set here. However, setting the values still prevents the
# questions
# from being shown, even if values come from dhcp.
d-i netcfg/get_hostname string unassigned-hostname
d-i netcfg/get_domain string unassigned-domain
d-i netcfg/wireless_wep string

### Mirror settings
# If you select ftp, the mirror/country string does not need to be set.
d-i mirror/country string enter information manually
d-i mirror/protocol string http
d-i mirror/http/hostname string at.archive.ubuntu.com
d-i mirror/http/directory string /ubuntu
d-i mirror/http/proxy string
# Suite to install.
d-i mirror/suite string precise

### Clock and time zone setup
d-i clock-setup/utc boolean true
d-i time/zone string Europe/Vienna
d-i clock-setup/ntp-server string 192.168.67.1

###
###    Partition
###

d-i partman-auto/disk string /dev/sda
d-i partman-auto/method string regular
d-i partman-auto/choose_recipe select atomic

d-i partman/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true

#
d-i partman/mount_style select uuid

# Grub installieren
d-i grub-installer/only_debian boolean true

### Base system installation
# The kernel image (meta) package to be installed; "none" can be used if no
# kernel is to be installed.
d-i    base-installer/kernel/altmeta string lts-saucy

### Apt setup
# You can choose to install non-free and contrib software.
D-i apt-setup/extras boolean true
d-i apt-setup/multiverse boolean true
d-i apt-setup/universe boolean true
d-i apt-setup/non-free boolean true
d-i apt-setup/contrib boolean true
d-i apt-setup/partner boolean true
d-i apt-setup/restricted boolean true
d-i apt-setup/backports boolean true
d-i debian-installer/allow_unauthenticated boolean true
```

```
# To create a normal user account.
d-i passwd/user-fullname string Administrator
d-i passwd/username string admin
d-i passwd/user-password-crypted password DAS_GEHEIME_PASSWORT
# keine Nutzungsinfos senden
popularity-contest popularity-contest/participate boolean false

### Package selection
## Desktop installieren
tasksel tasksel/first multiselect standard, ubuntu-desktop
# tasksel tasksel/first multiselect ubuntu-desktop
# d-i pkgsel/install-recommends boolean true
# d-i pkgsel/upgrade select full-upgrade
## deutsch und englisch
d-i pkgsel/language-packs multiselect de, en
## keine automatischen Updates
# d-i pkgsel/update-policy select none
# d-i pkgsel/updatedb boolean true

# Individual additional packages to install
d-i pkgsel/include string openssh-server joe nis nfs-common ntp wget gnome-shell gdm

#
# letzter command ein Script ausfuehren
#
d-i preseed/late_command string \
in-target wget http://192.168.67.1/scripts/init.sh ; \
in-target sh init.sh ; \
in-target rm init.sh ;

## Konsolenlayout
# d-i console-keymaps-at/keymap select de-latin1-nodeadkeys

### Finishing up the first stage install
# Avoid that last message about the install being complete.
d-i finish-install/reboot_in_progress note

xserver-xorg xserver-xorg/autodetect_monitor boolean true
xserver-xorg xserver-xorg/config/monitor/selection-method \
select medium
xserver-xorg xserver-xorg/config/monitor/mode-list \
select 1280x1024 @ 60 Hz
```

DAS_GEHEIME_PASSWORT muss mit dem Befehl

```
printf "secret_word" | mkpasswd -s -m sha-512
```

natürlich erst erstellt werden

Für eine Version in Englisch sind nur einige Kleinigkeiten zu ändern:

```
# locale
d-i debian-installer/locale string en_US.UTF-8
d-i languagechooser/language-name-fb select american
```

Diese (Enlische) Datei dann als **u12_04_en.cfg** auf ***/var/www*** speichern.

Die Desktop-Versionen der preseed Dateien haben alle Parameter gleich, bis auf:

```
d-i preseed/late_command string \
in-target wget http://192.168.67.1/scripts/init.sh ; \
in-target sh init.sh ; \
in-target rm init.sh ;
```

Diese Zeilen werden mit einem REM (#) versehen oder gelöscht.

> *Anmerkung:*
> *Der `preseed/late_command string` ermöglicht es zum Abschluss der Installation ein Shell-Script zu starten, dass weitere Arbeiten vornimmt. In meinem Fall ist das nur auf den Clients gewünscht, die ihre User-Laufwerke auf dem Server haben, zusätzliche Software brauchen etc.*

Diese Dateien werden als **u12_04_desk.cfg** b.z.w die Englische Version wieder als **u12_04_en_desk.cfg** gespeichert

Wie schon bereits erwähnt lade ich einen eigenen Kernel, der Eintrag

```
d-i base-installer/kernel/altmeta string lts-saucy
```

sorgt dafür, dass ein Kernel 3.11.0.17 (lts-saucy) geladen wird. Das ist in meinem Fall deshalb wichtig, da ich neue Hardware nutze und z.B. meine Netzwerkkarten mit dem LTS-Standardkernel nicht erkannt werden.

Die bootbare DVD

Zuerst erstellen wir ein paar Verzeichnisse in die Teile unserer DVD kopiert werden müssen.

```
mkdir -p /srv/tftp/Ubuntu/12.04/amd64
mkdir -p /srv/install/Ubuntu/12.04/amd64
```

Unter */srv/tftp* wird der Kernel abgelegt, unter */srv/install* werden die restlichen DVD-Dateien abgelegt, die für die Live-Version b.z.w Installation notwendig sind.

Dann laden wir uns das komplette ISO-File der Ubuntu 12.04 LTS Version in irgend ein Verzeichnis (`location/of/ISO`) mounten dieses ISO-File und kopieren einige Dateien.

```
mkdir /mnt/loop
mount -o loop -t iso9660 /location/of/ISO/ubuntu-9.10-desktop-amd64.iso /mnt/loop
cp /mnt/loop/casper/vmlinuz* /srv/tftp/Ubuntu/12.04/amd64
cp /mnt/loop/casper/initrd.lz /srv/tftp/Ubuntu/12.04/amd64
cp -R /mnt/loop/* /srv/install/Ubuntu/12.04/amd64
cp -R /mnt/loop/.disk /srv/install/Ubuntu/12.04/amd64
umount /mnt/loop
```

Unsere DVD sollten wir nun bereits per PXE starten können, für die Clients ist noch etwas Arbeit notwendig.

NIS für die Clients

NIS bietet eine zentrale Verwaltung der User und Gruppendateien an. Für kleinere Userzahlen ist NIS sicher eine gute Alternative, bei grossen Userzahlen wird man wohl auf LDAP umsteigen.

NIS ist hier auch deshalb eine gute Wahl, weil wir so die Useranlage unproblematisch per Script abwickeln können.

```
apt-get -y install nis
```

Auf dem Server muss NIS als Server gestartet werden, dazu muss die Datei ***/etc/default/nis*** editiert werden.

```
#
# /etc/defaults/nis     Configuration settings for the NIS daemons.
#

# Are we a NIS server and if so what kind (values: false, slave, master)?
NISSERVER=master

# Are we a NIS client?
NISCLIENT=false

# Location of the master NIS password file (for yppasswdd).
# If you change this make sure it matches with /var/yp/Makefile.
YPPWDDIR=/etc

# Do we allow the user to use ypchsh and/or ypchfn ? The YPCHANGEOK
# fields are passed with -e to yppasswdd, see it's manpage.
# Possible values: "chsh", "chfn", "chsh,chfn"
YPCHANGEOK=chsh

# NIS master server.  If this is configured on a slave server then ypinit
# will be run each time NIS is started.
NISMASTER=

# Additional options to be given to ypserv when it is started.
YPSERVARGS=

# Additional options to be given to ypbind when it is started.
YPBINDARGS=-no-dbus

# Additional options to be given to yppasswdd when it is started.  Note
# that if -p is set then the YPPWDDIR above should be empty.
YPPASSWDDARGS=

# Additional options to be given to ypxfrd when it is started.
YPXFRDARGS=
```

Weiters brauchen wir eine Domain, auf der NIS lauschen soll. Erinnern wir uns, was wir in die dhcp.conf eingetragen haben:

```
option nis-domain "ltsp.mydomain.tld";
```

Daher tragen wir nun in ***/etc/defaultdomain*** unsere Domain `ltsp.mydomain.tld` ein.

Damit NIS mit einheitlichen Daten arbeitet wird die Datei ***/etc/nsswitch.conf*** wie folgt angepasst:

```
passwd:         compat
group:          compat
shadow:         compat

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Der Befehl:

```
make -C /var/yp
```

müsste nun eigentlich ein Verzeichnis in ***/var/yp*** mit dem Namen unserer defaultdomain anlegen (also ltsp.mydomain.tld) und dort die (bereits vorhandenen) Userdaten ablegen.

!!!!!
Nach jeder Anlage eines Users muss der make-Befehl erneut gestartet werden.

Abschließende Arbeiten

Einige Arbeiten sind noch zu tun, so ist z.B. den Clients (während der Installation) noch „beizubringen“, die /home Laufwerke vom Server einzubinden, User per NIS abzufragen, zusätzliche Software bereitzustellen.

Springen wir gedanklich noch einmal zu unserer *preseed-Datei* und dort zu dem *late_command*.

```
d-i preseed/late_command string \
in-target wget http://192.168.67.1/scripts/init.sh ; \
in-target sh init.sh ; \
in-target rm init.sh ;
```

Bei der Installation wird die preseed-Datei komplett abgearbeitet, zum Schluss wird der genannte *late_command* ausgeführt. In unserem Fall wird das Shell-Script **init.sh** vom Webserver geladen. Da das Script sich im Unterverzeichnis ***/var/www/sripts*** befindet muss dieses Verzeichnis angelegt werden.

```
mkdir /var/www/scripts
mkdir /var/www/deploy
```

In diesem Verzeichnis muss ein Shell-Script mit dem Namen **init.sh** liegen. Mein init.sh-Script sieht wie folgt aus:

```
#
#
#      Clientinstallation
#
# alles nach tmp kopieren
cd /tmp

# holt aus dem Webverzeichnis deploy ein tar-file und entpackt es
wget http://192.168.67.1/deploy/install.tar.gz
tar -xzf install.tar.gz
rm install.tar.gz

# /home per nfs einbinden
#
echo "192.168.67.1:/home     /home   nfs    defaults        0       0\n" >> /etc/fstab

# Nis zugängig machen
#
echo "ypserver 192.168.67.1\n" >> /etc/yp.conf
echo "ltsp.mydomain.tld\n" > /etc/defaultdomain
echo "192.168.67.1    ltsp.mydomain.tld\n" >> /etc/hosts

# wlan ausschalten
#
echo "#!/bin/sh -e\n\n\nrfkill block wlan\nexit 0\n" > /etc/rc.local

# Usb deaktivieren
echo "blacklist usb-storage" >> /etc/modprobe.d/blacklist.conf

cp /tmp/nsswitch.conf /etc

#
#       Was sonst noch so zu machen ist.
#
```

Obiges Script holt u.A. ein tar.gz-File aus dem Verzeichnis /var/www/*deploy* in dem die Datei nsswitch.conf für unsere NIS-Anbindung der Clients liegt. Im tar.gz.File habe ich noch diverse Softwarepakete untergebracht, die zusätzlich auf dem Client installiert werden soll.

Das Script ist geeignet umfangreiche Änderungen am Client-System vorzunehmen. So kann z.B. weitere Software per apt-get installiert werden (ich installiere z.B. gdm und die gnome-shell und werfe Unity und lightdm raus) oder ein weiteres umfangreiches Shell-Script gestartet werden. Der Fantasie sind keine Grenzen gesetzt.

Die Datei **nsswitch.conf** für die Clients liest sich so:

```
passwd:         files nis
group:          files nis
shadow:         files nis

hosts:          files nis dns
networks:       files nis

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files
```

Diese Konfiguration sorgt dafür, dass Usereinträge zunächst in den lokalen Einträgen gesucht werden und sollten keine passenden Userdaten gefunden werden, wird auf den NIS-Server zurückgegriffen.

Ab hier sollte der Server voll Einsatzfähig sein. Alles was noch kommt ist genaugenommen Komfort b.z.w Feintuning und Kostmetik.

User einsperren

Für die Useranlage wird wie schon erwähnt ein Script (genauer sind es 2, von denen aber das Zweite vom Ersten aufgerufen wird) verwendet

Das eigentliche Anlagescript ist ein Perl-Script (***CreateUser.pl***):

```
#!/usr/bin/perl

use strict;

use File::Path;
use String::Random;

# ---------------------- Global constants ---------------------------------
my $AccountPrefix = "pc";
my $DomainSuffix = "\@ltsp.mydomain.tld";
my $DomainMailDir = "ltsp.mydomain.tld";
my $Separator = " ; ";
my $ProgramName = $0;
my $DomainMailSuffix = "\@".$DomainMailDir;

my $CreateNisEntry = "make -C /var/yp";
my @AccountSuffix = ("");
my @GroupRange = ("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
"21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "99");

# ---------------------- Output Files -------------------------------------
my $FileOut = $AccountPrefix."Accounts.csv";

# ---------------------- Commands -----------------------------------------
my $LinuxUserCmdLine="./addLinuxUser.sh"; # zweites Script aufrufen um User zu erstellen
my $MailUserCmdLine ="addmailuser"; # derzeit inaktiv, zum anlegen von Mailaccounts

# ---------------------- Begin Sources ------------------------------------
my $PasswdGen = new String::Random;

# Open local logfile
open(ACCOUNTFILE, ">$FileOut") or die "$ProgramName: File open failed for $FileOut $!: $@";

print ACCOUNTFILE "Nr;B1;P1;U1;M1;\n";
```

```
foreach my $group (@GroupRange)
{
   foreach my $accountSuffix (@AccountSuffix)
   {
      # Group Account Name
      my $accountName = $AccountPrefix.$group.$accountSuffix;
      # Group Account URI (email)
      my $accountURI = $accountName.$DomainMailSuffix;
      # Nitzschke am 11.10.2012
      my $sipaccoutURI = $accountName.$DomainSuffix;
      # Account Password Generate 8 character random password
      my $password = new $PasswdGen->randpattern("CCccnccn");
      my $mailpasswd = new $PasswdGen->randpattern("CCccnccn");

      my $printLine = $group.$Separator.$accountName.$Separator.$password.$Separator.$accountURI.$Separator.$mailpasswd.$Separator;

      print ACCOUNTFILE "$printLine";

#------------add Linux User-----------------
      my $addLinuxUserCmd = $LinuxUserCmdLine." ".$AccountPrefix.$group." ".$accountName." ".$password;
      print $addLinuxUserCmd."\n";
      `$addLinuxUserCmd`;

# ---------- Mailaccount -------------------
#      my $addmailUserCmd = $MailUserCmdLine." ".lc($accountName).$DomainMailSuffix." ".$mailpasswd;
#      `$addmailUserCmd`;

   }
   print ACCOUNTFILE "\n";
}
close(ACCOUNTFILE);

`$CreateNisEntry`;
```

Das Script erzeugt auf dem Server User mit dem Namen pc ($AccountPrefix) gefolgt von den Nummern 01 bis 35 ($GoupRange) plus einen User pc99 für den Vortragenden.

Über $AccountSuffix könnte noch ein Array definiert werden (z.B. „A“, „B“, „C“) so dass pc01, pc01A, pc01B und pc01C für mehrfache Zugänge generiert würden (z.B. pc01 für Netzwerk, pc01A für Grafik, pc01B für Programmieren etc).

Das vom Perl-Script aufgerufene Script ***addLinuxUser.sh*** sieht wie folgt aus:

```
#!/bin/sh

USERADD="/usr/sbin/useradd"
GROUPADD="/usr/sbin/groupadd"

group=$1
username=$2
pass=$3

       echo $pass > password
       ph=`makepasswd --clearfrom=password --crypt-md5|awk '{print $2}'`
       $USERADD -d /home/$username -m -p $ph -s /bin/bash $username

       usermod -G schulung $username

       # stiky bit und root als eigentümer, nur user darf noch zugreifen
       chown root.$username /home/$username
       chmod 1770 /home/$username

rm password
```

! Eventuell muss das Paket makepasswd noch installiert werden.

Entscheidend für das Einsperren sind hier das *chown* und das *chmod*. Es wird ein Sticky-Bit auf das Userverzeichnis gesetzt und der Eigentümer ist root. Der User hat also nur über die Gruppe Zugriff auf sein Verzeichnis. Jeder User wird der Gruppoe schulung hinzugefügt, so könnte z.B. ein gemeinsames Austausch-Verzeichnis mit entsrechenden Rechten angelegt werden.

Es gibt natürlich auch ein entsprechendes Script, dass die User wieder löscht (siehe am Schluss der Doku).

Backup

Das Backup ist ein Cronjob, der jede Nacht einmal läuft und mit rsync alle Userverzeichnisse sichert.

/script/make_backup.sh sieht wie folgt aus:

```
#!/bin/sh

echo `date` " backup started"  >> /backup/save/backup.log
rsync -av /backup/home/ /backup/save/ -R >> /backup/save/backup.log
echo `date` " backup completed "  >> /backup/save/backup.log
echo "----------------------------------------------"  >> /backup/save/backup.log
```

Der Croneintrag (crontab -e)

```
0 2 * * * /scripts/make_backup.sh
```

Zum Schluss die 3 Löschscripte. Zuerst ***DelUser.pl***

```
#!/usr/bin/perl

use strict;
use File::Path;

# --------------------- Global constants -------------------------------
my $AccountPrefix = "pc";
my $DomainSuffix = "\@ltsp.mydomain.tld";
my $DomainMailDir = "ltsp.mydomain.tld";
my $Separator = " ; ";
my $ProgramName = $0;
my $DelNisEntry = "rm /var/yp/".$DomainMailDir." -R";
my $DomainMailSuffix = "\@".$DomainMailDir;

my @AccountSuffix = ("");
my @GroupRange = ("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
"21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "99");
# --------------------- Commands ---------------------------------------
my $LinuxUserCmdLine="./delLinuxUser.sh"; # User löschen
my BackupHomeCmd="./rename_backup.sh"; # Das Backup vorher umbenennen
# --------------------- Begin Sources ----------------------------------

`$BackupHomeCmdLine`;

foreach my $group (@GroupRange)
{
   foreach my $accountSuffix (@AccountSuffix)
   {
      # Group Account Name
      my $accountName = $AccountPrefix.$group.$accountSuffix;
      # Group Account URI (email)
      my $accountURI = $accountName.$DomainSuffix;

      my $delLinuxUserCmdLine = $LinuxUserCmdLine." ".$AccountPrefix.$group." ".$accountName;
      print $delLinuxUserCmdLine."\n";
      `$delLinuxUserCmdLine`;
      print "$ProgramName: Successfully deleted account $accountName\n";

   }
}

`$DelNisEntry`;
```

Das obige Script ruft 2 weitere Scipte auf, `delLinuxUser.sh und rename_backup.sh`

delLinuxUser.sh:

```
#!/bin/sh

USERDEL="/usr/sbin/userdel"
GROUPDEL="/usr/sbin/groupdel"

group=$1
username=$2

              echo $USERDEL -r -f $username
              $USERDEL -r -f $username
              echo $GROUPDEL $group
              $GROUPDEL $group
```

löscht User, Verzeichnisse und Mailverzeichnisse.

rename_backup.sh:

```
#!/bin/sh

cd /backup/save
DATUM=$(date +"%Y_%m_%d_%H_%m")

dir="backup"

if [ -d $dir ]; then
mv $dir/ $DATUM/
fi
```

benennt vor dem Löschen der Daten im Home-Verzeichnis das Backup in ***Jahr_Monat_Tag_Stunde_Minute*** um, um sicher zu stellen, dass bei einer Neuanlage von Usern keine alten Daten überschrieben werden.

Um den Clients den Zugang zum Internet zu ermöglichen, muss ip_forward aktiviert werden und eine entsprechende Firewall-Regel generiert werden. Ein Listing meiner Firewallregeln:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.67.0/24      anywhere            ctstate NEW
ACCEPT     all  --  192.168.0.0/16       anywhere            ctstate NEW

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Um die Clients vom Internet zu trennen fahre ich einfach die Netzwerkkarte eth0 herunter.

Isch habbe Fertig!