Raspberry Pi als LTSP Thinclient verwenden

Der Raspberry Pi eignet sich mit einigen kleinen Einschränkungen ideal als LTSP-Client. Da der Raspberry Pi jedoch nicht über PXE booten kann, müssen die Clients ein kleines eigenes Image bekommen, das den Kernel und ggf. auch das ganze System bootet.

Dazu gibt es zwei verschiedene Ansätze:

Berry Terminal

Innerhalb der Berryboot Umgebung, einer Möglichkeit auf einfache Weise eine Multibootumgebung für den Raspberry einzurichten, gibt es ein fertiges Image Namens Berry-Terminal.

Diese stellt out of the Box einen ldm-Server bereit, der sich mit einer per dhcp aufgefundenen LTSP-Umgebung verbindet.

Bei Berry Boot ist das komplette Image auf der SD-Karte des Raspberries, was einige Einschränkungen mit sich bringt:

- keine lokalen Applikationen möglich
- Beim Umdate immer Update des Images nötig.
- keine zentrale Konfiguration über lts.conf

Nativer LTSP-Client

Alternativ kann auch eine LTSP-Umgebung für die armhf-Architektur wie gewohnt auf dem Server eingerichtet werden. Der einzige Unterschied ist, dass aufgrund der mangelnden PXE-Unterstützung, der Kernel auf der SD-Karte des Raspberries residieren muss.

Vorteil dieses Verfahrens:

- Lokale Apps möglich
- zentrale Konfiguration über die lts.conf im ltsp-chroot auf dem Server

Raspberry Pi Modelle

Version 1

Die hier beschriebene Anleitung funktioniert so weit - allein das Starten des ldm macht noch Probleme

🔧 Fix Me!

Version 2

Mit dem Raspberry Pi 2 gibt es noch Probleme augrund des fehlenden aufs Modules in dem Raspberry-Standard Kernel. (vgl.: https://pi-ltsp.net/advanced/kernels.html)

Hier muss ein eigener Kernel gebaut werden:

- http://rpitc.blogspot.co.uk/p/kernel-rebuild.html

Vorbereitung und Konfiguration

Hinweise und Links:

- http://cascadia.debian.net/trenza/Documentation/raspberrypi-ltsp-howto/
- https://pi-ltsp.net/
- https://raw.githubusercontent.com/PiNet/PiNet/master/pinet
- http://pinet.org.uk
- https://wiki.debian.org/DebianEdu/LTSPArm
- https://wiki.debian.org/RaspberryPi

Benötigte Packete:

```
apt-get install ltsp-server qemu-user-static binfmt-support ldm-server
```

Außer qemu-user-static und binfmt-support, die für die Cross-Plattform-Unterstützung nötig sind, sollte alles auf einem laufenden LTSP-Server bereits installiert sein.

Jetzt brauchen wir noch den gnupg-Key der Raspian Debian Binaries und müssen diesen in einen Keyring exportieren, der von der LTSP-Build-Umgebung genutzt werden kann:

Raspian-Gpg-Key suchen und importieren:

```
gpg --search-keys 90FDDD2E
# jetzt die Nummer des gefundenen Keys zum Importieren angeben
```

In einen extra Keyring exportieren:

```
gpg --export 90FDDD2E >> /etc/ltsp/raspbian.public.key.gpg
```

Wir benötigen noch eine eigene ltsp-Konfiguration mit Angaben für den speziellen Raspian Kernel und den Keyring:

/etc/ltsp/ltsp-raspbian.conf

```
DEBOOTSTRAP_KEYRING=/etc/ltsp/raspbian.public.key.gpg
DIST=wheezy
# For alternate raspbian mirrors,
# see: <http://www.raspbian.org/RaspbianMirrors>
MIRROR=http://archive.raspbian.org/raspbian
SECURITY_MIRROR=none
```

```
KERNEL_PACKAGES=linux-image-3.2.0-4-rpi
```

Der Raspberry Pi 2 mit Quadcore hat eine andere Prozessorarchitektur als das ältere Modell. Entsprechend muss für diese Geräte ein passender Kernel ausgewählt werden:

Z.B.:

```
KERNEL_PACKAGES=linux-image-3.18.0-trunk-rpi2
```

Client Image bauen und ggf. konfigurieren

Jetzt kann das Client Image wie gewohnt für LTSP gebaut werden:

```
ltsp-build-client --arch armhf --config /etc/ltsp/ltsp-raspbian.conf
```

Für den Fall, dass die ltsp-chroots per nfs exportiert werden (bei Debian Wheezy der default) und nicht squashfs-Images per nbd, liegt die `lts.conf` Konfiguration unter /opt/ltsp/ARCH/etc/lts.conf, wobei ARCH in diesem FAll mit armhf ersetzt werden muss.

Das bedeutet, das für jede Konfiguration eine eigene lts.conf existiert. Evtl. lässt sich das durch symlinks aber auch zentralisieren.

Kernel und Firmware auf die SD-Karte kopieren

Wir benötigen nun die Raspbian-Firmware Dateien. Dazu benötigen wir ein aktuelles Raspian Image, das wir als loopdevice mounten und die Firmwaredateien kopieren.

```
http://www.raspberrypi.org/downloads. Dieses Howto nutzt: 2015-02-16-
raspbian-wheezy.zip
```

Nach dem Download entpacken wir die Zip-Datei und Mounten Sie folgendermassen:

```
mkdir -p /tmp/rpi
mount -o loop,offset=$((512*8192)) 2013-02-09-wheezy-raspbian.img /tmp/rpi
```

Jetzt benötigen wir die SD-Karte für den Raspberry und mounten sie:

```
mount /dev/sdX1 /mnt
```

(sdX mit dem Device der SD-Karte ersetzen)

Kopieren der Firmware-Dateien aus dem Raspberry-Image:

```
cp -r /tmp/rpi/* /mnt/
```

```
umount /tmp/rpi
```

Und jetzt kopieren wir noch den Kernel und die Initrd:

cp -vb /opt/ltsp/armhf/boot/vmlinuz-3.2.0-4-rpi /mnt cp -vb /opt/ltsp/armhf/boot/initrd.img-3.2.0-4-rpi /mnt

Wir müssen noch die Datei `cmdline.txt` editieren, damit der Kernel die richtigen Kerneloptinen bekommt. Hier muss auch die IP-Adresse des LTSP-Servers angegeben werden:

Mit sed:

```
LTSP_SERVER=192.168.0.1 # durch eigene IP-Adresse ersetzen

sed -i -e 's,root=.*rootfstype=ext4,boot=nfs init=/sbin/init-ltsp
nfsroot='$LTSP_SERVER':/opt/ltsp/armhf,g' /mnt/cmdline.txt
```

Zuletzt editieren wir noch die config.txt, sozusagen das Bios des Raspberries, und geben hier den verwendeten Kernel und Initramdisk an (bei anderem Kernel entsprechend anpassen):

```
echo 'kernel=vmlinuz-3.2.0-4-rpi' >> /mnt/config.txt
echo 'ramfsfile=initrd.img-3.2.0-4-rpi' >> /mnt/config.txt
echo 'ramfsaddr=0x00800000' >> /mnt/config.txt
```

Für das Modell 2 muss als ramfsaddr folgendes eingetragen werden:

```
ramfsaddr=-1
```

(vgl. auch: https://github.com/raspberrypi/linux/issues/863)

Dieser Artikel steht unter der Creative Commons (BY-SA) Linzenz und darf unter gleichen Bedingungen weiter gegeben werden.

From:
https://wiki.datenkollektiv.net/ - **datenkollektiv.net**

Permanent link:
https://wiki.datenkollektiv.net/public/ltsp/raspberry

Last update: **2015/03/26 16:06**