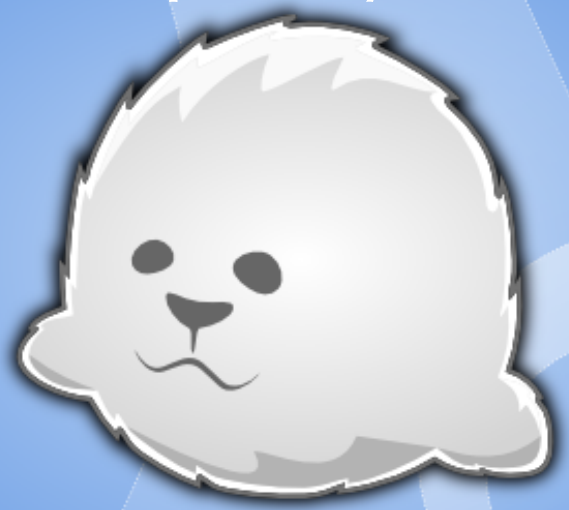
X2Go-ThinClientEdition-Live - jetzt auch für den Raspberry Pi?



Pi and More 12 ¼ – 2021

www.x2go.org

Vorstellung

Stefan Baur

Geschäftsführer der
BAUR-ITCS UG
(haftungsbeschränkt)

Vorstellung

Stefan Baur

X2Go-Projektkoordinator
X2Go-Eventplaner
X2Go-Lead-Evangelist

Was ist X2Go

- X2Go ist eine freie (F/LOSS) Remote-Desktop-/Remote-Application-Lösung
- Der Client ist für Linux, Windows und macOS verfügbar
- Der Client kann auch als GUI für RDP- (mit freeRDP oder rdesktop im Hintergrund) und XDMCP-Logins dienen
- Den Server gibt es aktuell nur für Linux

Was ist X2Go-TCE?

- X2Go-TCE ist ein standardisiertes, bootfähiges Image für diverse Bootmedien
- Herstellerübergreifend nutzbar, egal ob
  - dedizierter ThinClient (Igel, Rangee, Dell/Wyse, ...)
  - barebone (Shuttle, Zotac, Asus, Intel NUC, ...)
  - normaler, auch schon älterer PC
  - Neu: Raspberry Pi (ab Pi 3; Pi 4/Pi 400 empfohlen)

X2Go-TCE-Live oder X2Go-TCE-NFS?

- X2Go-TCE-NFS: diskless netboot (PXE+TFTP+NFS)
- X2Go-TCE-Live ist dagegen ein Livesystem (Ramdisk)
  - bootet auf Wunsch auch über das Netzwerk PXE+TFTP+HTTP(S) oder PXE+iPXE+HTTP(S)
  - aber alternativ auch von:
    - USB-Stick
    - SD-Card
    - fest eingebautem eMMC-Flashmedium
    - CD/DVD
    - HD/SSD

X2Go-TCE-Hardwareplattformen

- Praxiserprobt: x86/x64 (i386/i686; amd64)
- aktuell keine Relevanz (es gibt noch keine solche Clienthardware – ein Notebook damit befindet sich seit Jahren in der Entwicklung), aber vermutlich funktionsfähig: ppc64/ppc64le
- Jetzt neu: aarch64 in der Ausprägung Raspberry Pi
- Auf generischer aarch64-Hardware (also alles AUSSER dem Pi) vermutlich schon länger, weil man die ganzen Workarounds für den Pi nicht bräuchte – darauf will nur irgendwie keiner einen ThinClient haben ...

Hardwareprobleme bis Pi 3B/Pi 3B+

- PXE-Boot generell "hakelig", siehe:

  https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net.md
  → Abschnitt „Known problems“

- Bis einschließlich Pi 3B+ offene Bugs seitens Hersteller
  - „DHCP requests time out after five tries“
  - „DHCP request/reply/ack sequence not correctly implemented“
  - Restliche im Link genannten Probleme sind im 3B+ gefixt und nur im 3B nicht fixbar

X2Go-TCE auf dem Pi: Die Probleme

- Proprietäre Firmware muss geladen werden
- Debian Live/Live-Build sieht wohl den Raspberry Pi aktuell nicht vor → X2Go-TCE-Live läuft erst mal nicht
  - Wirft beim Build die Firmware nicht an die richtige Stelle → Boot Failed
  - Man muss sie daher „einsammeln“ und das Image damit patchen
- Auch auf dem aktuellsten Pi 4 dauern Builds sehr lange

  → besser nicht übertakten, wird ohnehin schon heiß

X2Go-TCE-Live auf dem Pi: HowTo

- Schmutzige Hacks beim Build (Module/Pakete injecten)
  - Aktualisiertes Buildscript, was auch für den Pi taugt, kommt voraussichtlich dieses Wochenende ins Wiki
  - Sofern ich noch was im Git-Repo aktualisieren muss, kommen diese Changes mit etwas Verzögerung im offiziellen Repo an → besser Linuxhaus-Repo wählen
- Beim Crossbuild auf x64 läuft das aarch64-Environment in einem Changeroot mit QEMU (nicht KVM)
  - aarch64-mksquashfs-Aufruf in QEMU sehr zäh
  - Trick: Skript benutzt stattdessen x64-mksquashfs

Wiki: X2Go-TCE-Live-Howto

- Hier findet ihr Buildscript und Config zum Copy-Pasten: https://wiki.x2go.org/doku.php/doc:howto:tce
- Bitte schaut zuerst im Skript, ob der folgende Textblock/Kommentar noch drin ist

```
[…]
# Note that ARM builds are currently not working, at least not
on the Pi.
[…]
```

→ Erst wenn er fehlt, sind die notwendigen Changes, damit der Raspi-Build funktioniert, im Skript enthalten

X2Go-TCE-Config: ARM-Builds

```
[…]
# Select ONE of the following LBX2GO_ARCH lines and comment out
the others
# (feel free to use long or short options)
# for 64-Bit builds, use:
# export LBX2GO_ARCH='-a amd64 -k amd64'
# 32-Bit, larger memory footprint, but faster performance on
i686 and newer
# export LBX2GO_ARCH='-a i386 -k 686-pae'
# 32-Bit, smallest memory footprint - not available on buster
# export LBX2GO_ARCH='--architectures i386 --linux-flavours 586'
# EXPERIMENTAL: For ARM (Raspberry Pi), try:
export LBX2GO_ARCH='-a arm64'
# Note that ARM builds are currently not working, at least not
on the Pi.
[…]
```

Linuxhaus-Repo

```
[…]
# Select ONE of the following git reposities

# this one loosely corresponds to "stable"

# export LBX2GO_CONFIG='git://code.x2go.org/live-build-
x2go.git::feature/openbox-magic-pixel-workaround-buster'

# this one loosely corresponds to "heuler"

export LBX2GO_CONFIG='https://github.com/LinuxHaus/live-build-
x2go::feature/openbox-magic-pixel-workaround-buster'

# NOTE: Add "-stretch" to the end of the LBX2GO_CONFIG string to
create a stretch build, and "-buster" for a buster build

# NOTE: As of 2019-08-27, buster builds are only available via
the github repo and for the feature/openbox-magic-pixel-
workaround-buster and feature/mate-minidesktop-buster branches
[…]
```

Dirty Patch is Dirty (I)

```
# after the build, let's determine the name of our image file …
IMAGEFILE="./x2go-tce-live-image-$(echo $LBX2GO_ARCH | \
   awk '{print $2}').img"
# ... and change the partition type to reflect the file system
# actually in use for partition 1
# ("b" is FAT32)
sfdisk --part-type $IMAGEFILE 1 b
# next, we need to patch two things inside the image, so we need
# to set up a loop device for it.
FREELOOP=$(losetup -f)
# note that this could become a TOCTOU issue if more than 1
# process tries to use loop devices

# as the image is a full disk image containing a partition, we
# need to jump to the position where the first partition starts
losetup -o 1048576 $FREELOOP $IMAGEFILE
# now let's mount it
mkdir -p ./tempmount
mount $FREELOOP ./tempmount
```

Dirty Patch is Dirty (II)

```
# purge this dir, so we have enough space; we'll return to fill
it later
rm ./tempmount/live/*
# first, we copy the contents of the boot/firmware/ folder to
the root directory, because that is where these files are needed
# see if inplace helps against out of space errors
rsync -aP --inplace ./chroot/boot/firmware/* ./tempmount
mkdir -p ./tempmount/live/
rsync -aP ./binary/live/*.squashfs ./tempmount/live/
# next, we replace the "root=" parameter with the parameters
needed for live-booting
sed -e 's#root=/dev/mmcblk0p2 #'"$LBX2GO_BOOTAPPEND_LIVE"' #' \
    -i ./tempmount/cmdline.txt
# here comes the cleanup part
sync
umount $FREELOOP
losetup -d $FREELOOP
rmdir ./tempmount
```

Dirty Patch is Dirty (III)

```
# modules required for Raspberry Pi 3 LAN
echo "crc16" >>
config/includes.chroot/etc/initramfs-tools/modules
echo "mii" >> config/includes.chroot/etc/initramfs-tools/modules
echo "smsc95xx" >>
config/includes.chroot/etc/initramfs-tools/modules
echo "usbcore" >>
config/includes.chroot/etc/initramfs-tools/modules
echo "usbnet" >>
config/includes.chroot/etc/initramfs-tools/modules
echo "fake-hwclock" >>config/package-lists/raspi.list.chroot
echo "usbutils" >>config/package-lists/raspi.list.chroot
# firmware for basic raspi functions - required for boot on Pi3
echo "raspi3-firmware/buster"
>>config/package-lists/raspi.list.chroot
# standard linux kernel - for Pi3
echo "linux-image-arm64/buster"
>>config/package-lists/raspi.list.chroot
```

Dirty Patch is Dirty (IV)

```
# firmware for wifi
echo "firmware-brcm80211/buster-backports" >>config/package-
lists/raspi.list.chroot
# firmware for basic raspi functions - required for boot on Pi4
echo "raspi3-firmware/buster-backports"
>>config/package-lists/raspi.list.chroot
echo "raspi-firmware/buster-backports"
>>config/package-lists/raspi.list.chroot
# newer linux kernel - required for pi4/pi400
echo "linux-image-arm64/buster-backports" >>config/package-
lists/raspi.list.chroot
```

Hacking chroot for fun and profit (I)

1) Wir forken eine Subshell in den Hintergrund, welche schaut, wann die Datei /chroot/usr/bin/mksquashfs angelegt wird
2) Sobald sie existiert, kopieren wir sie nach ./chroot/usr/bin/mksquashfs.real
3) Danach ersetzen wir ./chroot/usr/bin/mksquashfs durch ein Skript, was alle Aufrufparameter von ./chroot/usr/bin/mksquashfs nach /tmp/filesystem.squashfs.temp schreibt, und danach einfach nur wartet, bis dieses Tempfile wieder gelöscht wird

Hacking chroot for fun and profit (II)

1) Wir forken eine weitere Subshell in den Hintergrund, welche schaut, wann die Datei /tmp/filesystem.squashfs.temp angelegt wird
2) Sobald sie da ist, lesen wir die darin enthaltenen Parameter aus, passen sie an, und starten damit das native mksquashfs auf x86/x64
3) Sobald wir fertig sind, löschen wir /tmp/filesystem.squashfs.temp und moven ./chroot/usr/bin/mksquashfs.real zurück nach ./chroot/usr/bin/mksquashfs

Hacking chroot for fun and profit (III)

Pi4, 1.40GHz, 4 Cores, nativ, Sandisk Extreme Pro USB 3.1 (maximale Transferrate: 402 MByte/s)
real 25m7.527s
user 27m20.350s
sys 5m6.538s

i3-2100T, 2.50GHz, 4 Cores, QEMU-Changeroot, "spinning rust", 5400U/min, RAID1 (in LXC)

real 99m10.078s	real 60m1.724s
user 132m45.741s	user 54m57.586s
sys 4m35.599s	sys 4m57.304s
hack not enabled	hack enabled

i5-8250U, 1.60GHz, 8 Cores, QEMU-Changeroot, Disk s.u., (in einem VMware Gastsystem)

real 59m23,039s	real 36m45,144s	real 47m36,615s	real 25m36,343s
user 83m2,836s	user 29m43,260s	user 86m15,776s	user 29m48,128s
sys 7m12,040s	sys 6m43,255s	sys 6m15,318s	sys 6m7,810s
hack not enabled	hack enabled	hack not enabled	hack enabled
Extreme Pro USB 3.1	Extreme Pro USB 3.1	SATA-SSD 850 Evo	SATA-SSD 850 Evo

Laufzeit-Unterschiede:
Laufzeit des mksquashfs-Schritts: nativ amd64 ca. 4 Minuten, im QEMU-Changeroot über 26 Minuten

(alle Tests mit dem openbox-magic-pixel-workaround-buster-Branch und mit lokalem Paket-Cache (apt-cacher-ng))

X2Go-TCE auf dem Pi: Open Bugs

- Definitive Bugs:
  - Locale/Keymap wird offenbar noch nicht korrekt gesetzt
  - Nur Pi 3:
    - braucht momentan ein angepasstes Image
      Ziel ist: ein gemeinsames Image für Pi 3 & Pi 4
    - bootet nur sehr träge
- Ungetestet:
  - Audio
  - Druck- und Dateifreigaben

Nächster X2Go-Talk?

- Wir sind *vielleicht* bei den Chemnitzer LinuxTagen mit diesem Vortrag vertreten (Rückmeldung steht noch aus), auf jeden Fall aber mit einem virtuellen Projektstand: https://chemnitzer.linux-tage.de/2021/de
- Eventuelle Updates werden in den Vortrag für die CLT2021 eingepflegt und dort präsentiert
- Generell lohnt sich auch zu Covid-19-Zeiten ein Blick auf unsere Event-Seite im Wiki: https://wiki.x2go.org/doku.php/events:start

X2Go lebt vom Mitmachen!

- Helfer gesucht!
- Genau wie bei meinem anderen Projekt, dem Grannophone, gilt, dass X2Go immer zwei Dinge von euch brauchen kann:
  - Zeit/KnowHow – auch von Nicht-Programmierern!
  - Geld/Hardware/Dienstleistung: Man kann ...
    - über den orca e.V. (gemeinnützig) eine zweckgebundene Spende an X2Go leisten
    - eine der Firmen im Projekt mit einer konkreten Aufgabe (Bugfix, Feature Request) beauftragen

Spenden/Sponsoring/Aufträge

- Für Spenden bitte mit dem orca e.V. Kontakt aufnehmen: https://orca-ev.de/ (Spendenseite wird demnächst online gestellt, vorerst bitte Mail senden)
- Für Sponsoring/Aufträge stehen die Firmen unter https://wiki.x2go.org/doku.php/0spnn5 (Null-spnn-Fünf) zur Verfügung.

Aktuelle Sponsoren der
X2Go-TCE-Pi-Portierung:
Rick & Andrew Gregory
von gbgsoft.com

http://gbgsoftdev.blogspot.com/p/about-us.html

Vielen Dank für euer Interesse!