Inhaltsverzeichnis

Einleitung

I-Tools ist eine Sammlung von Hilfsprogrammen, für die Behandlung von Image-Dateien vornehmlich für embedded Linux-Devices, zum Beispiel Raspberry Pi, Banana Pi, Beaglebone, etc.

Die Beispiele in diesem Dokument beziehen sich hauptsächlich auf den Raspberry Pi, da dieser zur Zeit am populärsten ist. Generell eignet sich die Programmsammlung I-Tools auch für andere embedded Linux-Devices oder für Industrie-PC's (IPC) mit einem Linux-Betriebssystem.

Dabei handelt es sich um folgende Einzelprogramme:

- **iwrite** Dieses Programm dient als Ersatz von „**dd**“. Hiermit kann eine Image-datei auf eine SD-Karte, CompactFlash, USB-Stick, o.ä, kopiert werden. Eine versehentliche Fehleingabe, z.B. /dev/sda, die die Daten der Festplatte des Arbeitsrechners unwiederbringlich beschädigen würde, ist dabei ausgeschlossen, ferner wird beim Kopiervorgang ein Fortschrittsbalken angezeigt.
- **iread** Das Gegenstück zu **iwrite**, der Inhalt eines externen Datenträgers wird unter den gleichen Sicherheitsvorkehrungen in eine Imagedatei geschrieben. Der Kopiervorgang wird ebenfalls mit einem Fortschrittsbalken angezeigt. Es werden – falls nicht anders gewollt - nur die gefundenen Partitionen kopiert und nicht der Inhalt des gesamten Datenträgers.
- **imount** Mit diesem Programm können einzelne oder alle Partitionen einer Imagedatei oder eines externen Speichermediums (Block-device) an einer beliebigen Stelle eingehängt (gemounted) oder wieder ausgehängt werden.
- **cross-chroot** Mit diesem Programm kann eine Chroot-Shell unter der Verwendung von **qemu**, oder gezielt Programme auf einer Imagedatei oder eines externen Speichermediums auf dem PC ausgeführt werden. Damit ist es z.B. möglich auf einen x86-Linux-PC Linux-Programme die z.B. für einen ARM-Prozessor compiliert worden sind, auszuführen. Somit können z.B. Programme für den Raspberry Pi oder Beagle-Bone, etc.., auf dem PC ausgeführt werden.
- **iparted** Erweiterungsscript für das Programm **gparted**. Hiermit können Imagedateien und die darin befindlichen Partitionen vergrößert, verkleinert, angelegt oder verschoben werden. Eine versehentliche Fehleingabe, welche fatale Folgen für die Daten der Festplatte haben könnte, ist ausgeschlossen.
- **rellink** Hiermit können in einem Verzeichnis und (falls gewünscht) dessen Unterverzeichnissen symbolische Links, die auf eine Orginaldatei zeigen, welche sich im selben Verzeichnis befindet, relativiert werden. Eine Relativierung symbolischer Links, wird notwendig, wenn z.B. ein Cross-Compiler auf Librarys zurückgreift, die auf einem Pseudo-Root-Linux-Verzeichnis, des nativen Arbeitsrechners befinden.

- **xsudo** Erweiterungsscript welches **sudo** aufruft für den Aufruf von Programmen mit grafischer Oberfläche (X11), bei denen Administratorrechte (root) erforderlich sind.

Zusätzlich benötigte Programme

Für die Verwendung der I-Tools wird die Installation zusätzlicher Programme Vorausgesetzt, welche – sofern sie nicht schon installiert sind – durch entsprechende Installationsprogramme der jeweiligen Linux-Distribution nachinstalliert werden können oder – falls von der Distribution nicht angeboten - von der entsprechenden Webside heruntergeladen werden können.

pv (Pipe-viewer)

URL:

http://www.ivarch.com/programs/pv.shtml

Version >= 1.5.7

Monitorprogramm zur Darstellung von Daten in einer Pipe mit Fortschrittsbalken.

Dieses Programm ist obligatorisch für die Programme **iwrite** und **iread.**

Xdialog

URL:

http://xdialog.free.fr/

Version >= 2.3.1

Programm zur Darstellung von grafischen Dialogboxen (X-Windows)

Dieses Programm ist für **iparted** obligatorisch, für **iwrite** und **iread** optional.

chroot

URL:

http://ftp.gnu.org/gnu/coreutils/

Programm zum Wechseln des Root- Verzeichnisses im aktuellen Prozess und seiner Kind-Prozesse.

Dieses Programm ist für **cross-chroot** obligatorisch.

qemu

URL:

http://wiki.qemu.org

Quick-Emulator. Programm zur Emulation von Fremdprozessoren (z.B. ARM) auf dem Arbeits-PC.

Dieses Programm ist für **cross-chroot** obligatorisch.

gparted (Gnome Partition Editor)

URL:

http://gparted.org

Version >=0.14.1

Grafischer Editor zur Behandlung von Festplatten-Partitionen.

Dieses Programm ist für **iparted** obligatorisch.

Installation

Die Datei /etc/dev_permissions.conf

Zentraler Bestandteil der Programmsammlung I-Tools ist die Konfigurationsdatei **`dev_permissions.conf`**, die sich im zentralen Verzeichnis für Konfigurationen **`/etc/`** befindet. Da einige Programme von I-Tools auf diese Datei zugreifen, funktioniert ohne **`dev_permissions.conf`** so gut wie nichts. Und das ist gut so!

In dieser Datei sind die Gerätedateien (Block-Devices) aufgelistet, die beim Anschluss von einen oder mehreren externen Datenträgern mit diesen verknüpft werden. Jedes mal wenn eines der Programme von I-Tools schreibend oder auch nur lesend auf eine Gerätedatei zugreifen möchte, wird der Gerätedateiname mit den eingetragenen Namen in der Datei `dev_permissions.conf` verglichen. Falls dieser Name nicht in der `dev_permissions.conf` eingetragen ist, wird die Programmausführung abgebrochen.

Auf keinen Fall dürfen in dieser Datei die Gerätedateien stehen, welche mit den Festplatten des PS's verknüpft sind! Denn somit wäre das Sicherheitskonzept der I-Tools nicht mehr gegeben! Zum Beispiel sollte „/dev/sda“ **nicht** in dieser Datei stehen, weil diese Gerätedatei in der Regel die erste physikalische Festplatte im PC repräsentiert! „/dev/sdb“ würde die zweite physikalische Festplatte (falls vorhanden) repräsentieren, usw. Daher werden sich auch die Gerätedateinamen für externe Datenträger je nach Anzahl der fest installierten Festplatten ändern.

Da jeder PC von seiner Struktur der Festplatten individuell verschieden sein kann – jeder PC kann eine oder mehrere physikalische Festplatten enthalten – muss die Datei `dev_permissions.conf` individuell an die Beschaffenheit des jeweiligen PC's angepasst werden. Das wird auch jedes mal wieder erneut notwendig werden, wenn weitere zusätzliche Festplatten eingebaut, getauscht oder ausgebaut werden.

Aus diesem Grund ist die Datei `dev_permissions.conf` nicht im Lieferumfang enthalten, sondern muss mittels den Programm **mk-permit** erzeugt oder mit einen beliebigen Text-Editor manuell erstellt werden.

Automatisches Erstellen der Datei dev_permissions.conf

1)

Entfernen Sie zunächst alle externen Wechsel-Datenträger, falls welche angeschlossen sind, von Ihrem PC. Wechseldatenträger, welche nicht entfernt werden, werden sonst nicht in die Datei dev_permissions.conf als erlaubt aufgenommen und sind somit für die Programme von I-Tools ebenfalls gesperrt.

2)

Melden Sie sich als Administrator Root an und starten sie in einer Shell das Programm **mk-permit** und folgen Sie den Anweisungen.

```
~> sudo mk-permit
```

oder als Root:

```
~# mk-permit
```

3)

Bei erfolgreicher Ausführung hat das Programm **mk-permit** die Datei **dev_permissions.conf** im Verzeichnis **/dev/** angelegt.

Öffnen sie zur Sicherheit die Datei /dev/dev_permissions.conf mit einen beliebigen Taxteditor oder Betrachter und vergewissern Sie sich ob dort wirklich nur die Gerätedateien angegeben sind, welche mit externen Datenträgern verknüpft werden.

Dieser Vorgang muss unbedingt wiederholt werden, falls Änderungen an den physikalischen Festplatten vorgenommen worden (z.B Einbau oder Ausbau einer weiteren Festplatte).

Erstellen der Datei dev_permissions .conf per Hand

1)

Entfernen Sie zunächst alle externen Wechsel-Datenträger, falls welche angeschlossen sind, von Ihrem PC.

2)

Öffnen Sie eine beliebige Textkonsole Ihrer Wahl und geben sie folgenden Befehl ein[1]:

```
~> df -h | grep "/dev/sd[a-z]"
```

Dieses Kommando führt in diesem Beispiel zu folgender Ausgabe:

```
/dev/sda6        20G     18G  1,4G   93% /
/dev/sdb1       458G    212G  223G   49% /hdd/v1
/dev/sda1       233G     88G  146G   38% /windows/C
/dev/sda7       207G    131G   66G   67% /home
```

Hier ist zu sehen, dass in diesem Beispiel-PC zwei physikalische Festplatten angeschlossen sind, nämlich `/dev/sda` und `/dev/sdb` die abschließende Nummer gibt die jeweilige Partitionsnummer an. Am Schluss der Zeilen wird jeweils der Einhängepunkt (mountpoint) der betreffenden Partition gezeigt. Zum Beispiel befindet sich die Root-Partition auf /dev/sda6.

ACHTUNG: Hieraus entnehmen wir die Information, dass in diesem Fall `/dev/sda` und `/dev/sdb` unter keinen Umständen in die Datei `dev_permissions.conf` eingetragen werden dürfen! Diese sind ein absolutes Tabu!

Somit wäre die nächste freien Gerätedateien (Block-Devices), welche mit einen externen (USB-) Datenträger, verknüpft werden können theoretisch /dev/sdc, /dev/sdd, /dev/sde, … usw. Allerdings sollten wir diese auch nicht blind in die Datei `dev_permissions.conf` eintragen, sinnvoll ist es, das vorher nochmal praktisch zu überprüfen.

3)

Verbinden Sie die/den externen Datenträger SD-Karte(n), CompactFlash(s) USB-Stick(s)... usw. mit Ihrem PC. Wichtig dabei ist, dass der/die Datenträger auch im System eingehängt

1 Sie können den Befehl ab „df“ via copy & paste aus diesem Dokument in die Eingabekonsole kopieren. Die Pipe zu „grep“ filtert die Informationen heraus, die für uns interessant sind.

(gemounted) ist/sind, da er/sie sonst von **df** nicht erkannt werden. Bei den gängigen grafischen Linux-Desktops (KDE, GNOME, LXDE, Xfce) können diese über den Gerätemanager mit einem Mausklick eingehängt und wieder ausgehängt werden.

4)

Wiederholen Sie jetzt nochmal den oben stehenden Befehl:

```
~> df -h | grep "/dev/sd[a-z]"
```

Falls die angeschlossenen externen Datenträger erfolgreich eingehängt sind, könnte in Etwa folgende Ausgabe erscheinen:

```
/dev/sda6        20G     18G  1,4G   93% /
/dev/sdb1       458G    212G  223G   49% /hdd/v1
/dev/sda1       233G     88G  146G   38% /windows/C
/dev/sda7       207G    131G   66G   67% /home
/dev/sde2       2,7G    2,0G  569M   79% /run/media/uli/fc254b57-8fff-4f96-9609-ea202d871acf
/dev/sde2       2,7G    2,0G  569M   79% /var/run/media/uli/fc254b57-8fff-4f96-9609-ea202d871acf
/dev/sde1        56M    9,5M   47M   17% /run/media/uli/boot
/dev/sde1        56M    9,5M   47M   17% /var/run/media/uli/boot
/dev/sdc3       7,1G    329M  6,4G    5% /run/media/uli/9c8ed414-669c-445b-a14f-d24184ae9847
/dev/sdc3       7,1G    329M  6,4G    5% /var/run/media/uli/9c8ed414-669c-445b-a14f-d24184ae9847
/dev/sdc2       5,7G    3,9G  1,6G   71% /run/media/uli/6f23d6e9-41a8-492f-a1d2-755b4ab5bd00
/dev/sdc2       5,7G    3,9G  1,6G   71% /var/run/media/uli/6f23d6e9-41a8-492f-a1d2-755b4ab5bd00
```

In diesen Beispiel wurden zwei externe Datenträger (eine SD-Karte und ein ConpactFlash) mit jeweils zwei Partitionen eingehängt. Neben den beiden Festplatten-Partitionen (/**dev/sda**x und **/dev/sdb**x) werden jetzt auch zusätzlich die Partitionen von **/dev/sdc**x (ein ConpactFlash für einen Industrie-PC) und **/dev/sde**x (ein Image für den Raspberry Pi) angezeigt. Sollte ein frisch gekaufter mit FAT32 vorformatierter Datenträger eingehängt sein, so würde **df** z.B. nur /dev/sde anzeigen ohne die Partitionsnummer da keine Partitionen vorhanden sind.
Somit sind die Gerätedateien **/dev/sdc** und **/dev/sde** mit den externen Datenträgern verknüpft, und können somit in die Datei `dev_permissions` geschrieben werden.

5)
Öffnen Sie als Administrator (als root einloggen oder via su) einen beliebigen Texteditor und legen Sie die Datei **`dev_permissions.conf`** an.
Tragen sie die beiden voll qualifizierten Gerätedateinamen in die Datei ein und speichern Sie diese in dem Verzeichnis **`/etc/`**. Die folgenden Zeilen zeigen den Inhalt dieser Datei, wie sie anhand unseres Beispiels aussehen sollte (natürlich ohne die Kommentare):

```
# Das ist ein Kommentar!
/dev/sdc
# /dev/sdd
/dev/sde # Das ist noch ein Kommentar!
```

Zur Anschaulichkeit ist hier auch noch die Gerätedatei /dev/sdd eingetragen, jedoch ohne Wirkung, da sie mittels dem Hash-Zeichen # auskommentiert ist. Das Hash-Zeichen leitet bei Linux-Konfigurationsdateien üblicherweise einen Kommentar ein, der bis zum

Zeilenende seine Gültigkeit hat.

iwrite (Image-Writer)

Das Programm **iwrite.sh** dient in erster Linie dazu den Programmaufruf, für das in die Jahre gekommene UNIX-Kopierprogramm „**dd**“, zu ersetzen. **dd** stammt aus den 1970-Jahren zu einer Zeit, wo Dateien im Byte oder Kilobyte-Bereich kopiert wurden. Megabyte, Gigabyte oder Terabyte waren zu dieser Zeit noch Begriffe aus dem Sciencefiction.

Des weiteren ist dieser Befehl auch sehr gefährlich, da für das Kopieren einer Imagedatei auf ein Block-Bevice (z.B. einer SD-Karte) Administratorrechte (Root-Rechte) erforderlich sind. Eine falsche Angabe des dd-Parameters „of=/dev/sdx“ – z.B. ein Tippfehler aus Übermüdung – kann zu irreparablen Schäden der Daten auf der Festplatte des Arbeitsrechners führen. Zum Beispiel wenn statt of=/dev/sdc *nur* of=/dev/sda (/dev/sdc sei hier der SD-Kartenleser) eingetippt wird, also versehentlich bei dem letzten Buchstaben versehentlich „a“ statt „c“ angibt, wird das sehr wahrscheinlich zu schwerwiegenden Datenschäden auf der ersten Festplatte führen, welche in der Regel durch die Gerätedatei „/dev/sda“ im System repräsentiert wird.

Ferner ist das Kopieren einer mehrere Gigabyte großen Datei auf eine SD-Karte, CompactFlash oder USB-Stick sehr zeitintensiv. Das Kopieren einer 8 Gigabyte großen Datei kann gut und gerne 25 Minuten oder noch länger dauern! **dd** würde wehrend dieser Zeit keinerlei Aktionen auf dem Bildschirm zeigen, es entsteht der Eindruck, als wäre der Prozess abgestürzt. Lediglich die Status-LED vom SD-Kartenleser (falls vorhanden) würde ab und zu blinken.

Bei **iwrite.sh** ist eine versehentliche Fehleingabe ausgeschlossen[2] in diesem Fall würde des Programm mit einer Fehlermeldung beendet.

Wahrend des Kopiervorgangs mittels **iwrite** wird – je nach gesetzter Option – ein Fortschrittsbalken in einer X-Windows-Dialogbox oder einer textbasierenden Dialogbox oder in Form von „=“ - Zeichen angezeigt.

Falls bei iwrite.sh kein Zielgerät als zweiter Parameter angegeben wird, sucht das Programm – bei Option -d oder -x – automatisch nach Zielgeräten, die dafür in Frage kommen könnten, wird mehr als ein Zielgerät gefunden, erscheint ein entsprechendes Auswahlmenü. Es werden nur die Zielgeräte gefunden, die von ihrer Speicherkapazität ausreichend groß für die auf der Image-Datei befindlichen Partitionen sind und natürlich in der Datei /etc/dev_pernissions.conf eingetragen sind.

Was macht iwrite?

- Zuerst werden mögliche Optionen ausgewertet.
- Falls keine weiteren Aufrufparameter übergeben werden, erscheint im Dialogbetrieb – (bei Option „-d“ oder „-x“) – ein Datei-Auswahlmenü, um eine Imagedatei auszuwählen. Das Dateimenü erscheint auch, wenn als erster Parameter statt einem voll qualifizierten Dateinamen (Pfadangabe/Dateiname) nur eine Pfadangabe gemacht wird, in diesem Fall werden im Dateimenü gleich die Dateien im angegebenen Pfad gezeigt.
- Die maximale Anzahl der zu kopierenden Bytes werden anhand der auf der

2 Vorausgesetzt die Datei /etc/dev_permissions.conf stimmt.

Imagedatei gefundenen Partitionen berechnet, falls keine Partitionen gefunden werden oder die Option „-e, --entire“ gesetzt wurde, wird als Anzahl die volle Dateigröße verwendet.

- Falls der zweite Parameter fehlt, sucht **iwrite** bei allen in der Datei `/etc/dev_permissions.conf` angegebenen Gerätedateien nach einem passenden Speichermedium, was von der Speicherkapazität als Ziel in Frage kommt. Wird mehr als ein Speichermedium gefunden, erscheint im Dialogbetrieb (Optionen „-d“ oder „-x“) eine Auswahldialogbox mit den Gerätedateien und der Speicherkapazität der damit verknüpften Speichermedien, welche als Ziel in Frage kommen. Andernfalls wird das Programm einer entsprechenden Fehlermeldung abgebrochen.
- Falls der zweite Parameter angegeben wird, der aus dem voll qualifizierten Namen der Gerätedatei, welche mit dem Zielgerät verknüpft ist besteht, oder nur aus dem letzten Buchstaben z.B. „/dev/sde“ oder nur „e“ (für MS-Windows-Nostalgiker), wird diese Eingabe mit der Datei `/etc/dev_permissions.conf` verglichen, falls in dieser Datei kein entsprechender Eintrag gefunden wird, bricht das Programm seine Ausführung mit einer entsprechenden Fehlermeldung ab. Sollte der zweite Parameter aus einem symbolischen Link auf eine Gerätedatei bestehen wird dieser vor dem Vergleich von **iwrite** dereferenziert. Anschließend wird geprüft, ob die Speicherkapazität des angegebenen Zielgerätes ausreicht, andernfalls wird das Programm mit einer entsprechenden Fehlermeldung beendet.
- Falls nicht Option -Y, --yes nicht gesetzt wurde, wird nun die Integrität der Imagedatei (der Datenquelle) überprüft. Das heißt es wird untersucht, ob es sich bei der Datei wirklich im ein Festplattenabblid handelt, indem untersucht wird, ob eine Partitionstabelle vorhanden ist. Falls nicht, wird eine entsprechende Rückfrage gestellt, ob die Programmausführung beendet werden soll oder nicht.
- Falls nicht Option -Y, --yes nicht gesetzt wurde, erscheint eine Sicherheitsabfrage, welche nochmal die Datenquelle, also die Imagedatei, und die Datensenke, also das Zielspeichermedium anzeigt. Wird diese Frage nicht positiv quittiert, bricht das Programm seine Ausführung ab.
- Als nächstes prüft **iwrite**, ob des Zielspeichermedium irgendwo im System eingehängt (gemounted) ist. Ist das der Fall, wird versucht das Zielspeichermedium auszuhängen (umount). Falls das nicht gelingt, was daran liegen könnte, dass irgend ein Programm noch auf dieses Medium zugreift (z.B. der Dateimanager), wird die Programmausführung mit einer entsprechenden Fehlermeldung beendet.
- Es wird ermittelt, ob bei dem Zielspeichermedium (dem externen Datenträger) ein eventueller Schreibschutz (bei SD-Karten der kleine Schiebeschalter an der Seite) aktiviert ist. Trifft das zu, wird die Programmausführung mit der entsprechenden Fehlermeldung beendet.
- Es wird untersucht, ob die Imagedatei irgendwo im System eingehängt (gemounted) ist. Trifft das zu, versucht **iwrite** via **imount -u** die Imagedatei auszuhängen. Bei Erfolg merkt sich das Programm den Einhängepunkt um die Imagedatei nach Abschluss des Kopiervorgangs wieder einzuhängen. Im Fehlerfall, was daran liegen könnte, dass irgend ein anderes Programm (z.B. der Dateimanager) noch auf eine oder mehrere Partitionen der Imagedatei zugreift, wird

das Programm mit einer entsprechenden Fehlermeldung beendet.

- Der eigentliche Kopiervorgang beginnt. Dieser kann einige Minuten in Anspruch nehmen. Bei einem Datenvolumen von 8 GB kann dieser Vorgang, je nach Kartentyp, um die 25 Minuten dauern. Wenn die Option -d gesetzt wurde wird der Kopiervorgang in einer pseudografischen Dialogbox als Fortschrittsbalken angezeigt. Bei der Option -x erscheint der Fortschrittsbalken in einer X-Windows-Dialogbox. Ansonsten wird er Fortschrittsbalken auf der Textkonsole mit „=“ angezeigt. (siehe Abbildungen bei den Beispielen).
- Nach Abschluss des Kopiervorgangs versucht **iwrite** die Imagedatei wieder an dem gemerkten Einhängepunkt via **imount** einzuhängen, falls sie von **iwrite** vorher ausgehängt wurde.

Aufruf von iwrite

Aufruf:

```
iwrite.sh [options] [Source-Imagefile] [Target Block-Device]
```

Option -h, --help

Anzeige der Hilfe.

Option -v

Ausgabe von zusätzlichen Informationen auf der Textkonsole.

Option -s, --silence

Es wird kein Fortschrittsbalken angezeigt nach der Sicherheitsüberprüfung wird für den Kopiervorgang lediglich **dd** aufgerufen.

Option -d, --dialog

Dialogbetrieb in pseudografischen Dialogboxen.

Option -x, --xdialog

X-Windows Dialogbetrieb.
Falls der Programmaufruf im Usermodus via **sudo** nicht funktioniert, sollte statt dessen **xsudo.sh** verwendet werden, oder man muss sich vorher via **su** als root anmelden.

Option -l

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf` angezeigt.

Option -c

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf`

angezeigt, die zur Zeit mit einem physikalischen Speichermedium verknüpft sind.

Option -e, --entire

Wenn diese Option gesetzt wurde, wird die die komplette Imagedatei auf des Speichermedium kopiert. Ansonsten werden nur die Daten bis zur Obergrenze der höchsten gefundenen Partition kopiert, falls welche gefunden wurden.

Diese Option ist nur sinnvoll, wenn sich noch weitere Daten auf der Imagedatei befinden sollten, die oberhalb der höchsten Partition liegen. Normalerweise sollte die Dateigröße der Imagedatei und die Obergrenze der höchsten Partition identisch sein.

Änderungen, der in der Imagedatei befindlichen Partitionen, können mit dem Programm **iperted.sh** gemacht werden.

Option -n

Es findet findet kein Kopiervorgang bzw irgendwelche Scheribvorgänge auf dem Zielgerät statt. Diese Option dient für Testzwecke und der Programmwartung.

Option -Y, --yes

Diverse Sicherheitsabfragen werden übersprungen. bzw. automatisch positiv quittiert. Diese Option ist nur wirksam, wenn das Zielspeichermedium beim Programmaufruf explizit im zweiten Parameter angegeben wurde.

Option -L=<RATE>

Limitiert die Übertragungsrate für das Programm „**pv**“ welches – wenn Option „-s“ nicht gesetzt wurde – den Kopiervorgang ausführt, in Bytes pro Sekunde. Die Einheit "k" für Kilobyte , "m" für Megabyte, "g" für Gigabyte, oder "t" für Terabyte kann hinter die Zahl gesetzt werden. Der Standardwert ist auf 5m, also 5 Megabyte pro Sekunde, eingestellt. Falls die Anzeige vom Fortschrittsbalken stark von tatsächlichen Kopierfortschritt differiert, z.B. wenn der Fortschrittsbalken schon bei 100% steht, aber der eigentliche Kopiervorgang erst nach einer deutlichen Verzögerung abgeschlossen ist, macht es Sinn eventuell mit dieser Option einen anderen Wert auszuprobieren.

Beispiele

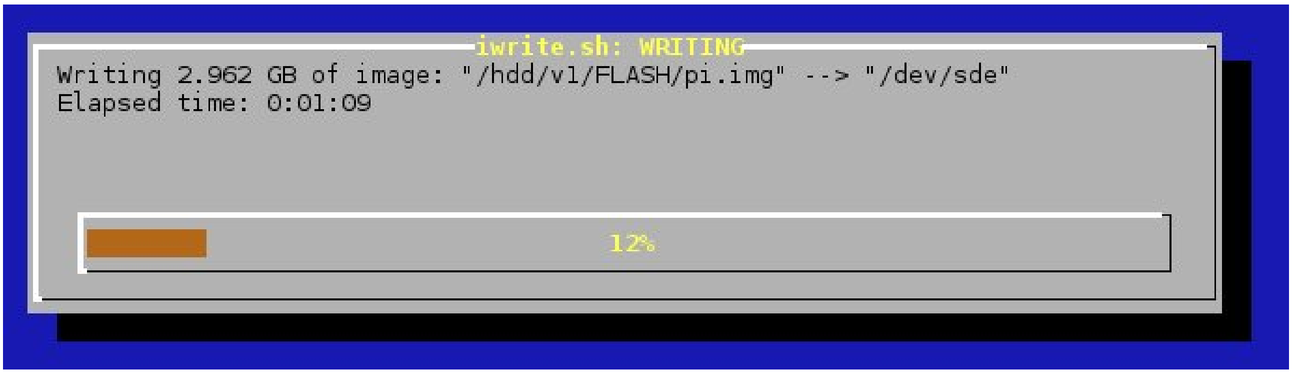


Abbildung 1: Fortschrittsbalken bei der Option "-d"

Abbildung 2: Fortschrittsbalken bei der Option -x

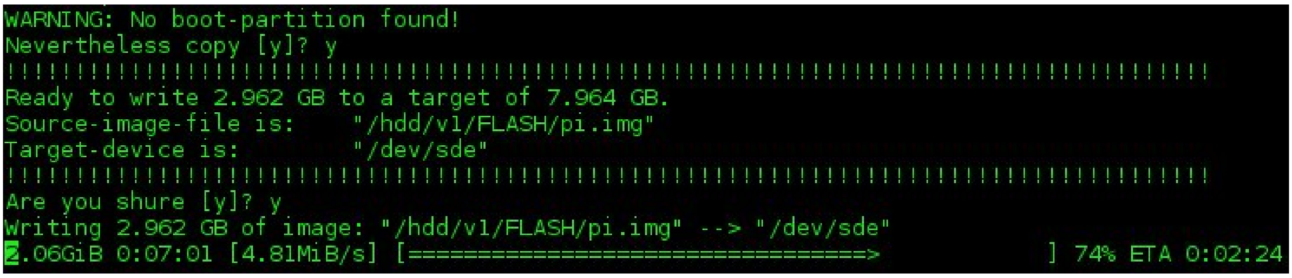


Abbildung 3: Fortschrittsbalken wenn weder die Option -d noch -x und -s gesetzt wurden

Abbildung 4: Auflistung angeschlossener Speichermedien bei den Optionen -xc

iread (Image-Reader)

Des Programm **iread.sh** ist das Gegenstück zu dem Programm **iwrite.sh**, hiermit können alle Daten auf einem externen Datenträger (SD-Karte, USB-Stick, CompactFlash, etc..) zurück in eine Imagedatei geschrieben werden. Dabei wird – falls Option -e, --entire nicht gesetzt wurde – nicht der komplette Speicherinhalt von dem Datenträger kopiert, sondern nur alle darauf befindlichen Partitionen, falls welche gefunden wurden. Dies kann den Kopiervorgang ggf. deutlich beschleunigen, da kein Datenmüll oberhalb der höchsten Partition sinnlos in die Zieldatei kopiert wird, wie das bei dem Aufruf von **dd** der Fall wäre.

Gesetzt dem Fall man würde zum Beispiel eine SD-Karte für den Raspberry-Pi vorbereiten, die ein Datenvolumen von 8 GB hat. Z.B. mittels den Programm **iwrite.sh** kann eine vom Internet heruntergeladene Imagedatei, die entpackt eine z.B. eine Größe von nur 3 GB hat, auf die SD-Karte kopiert werden. Falls man nicht später auf dem Rasbery-Pi mittels dem Programm **raspi-config** die Root-Partition auf des volle Datenvolumen von 8 GB expandiert[3], bleiben die oberen 5 GB unbenutzt. Würde man jetzt den Inhalt der SD-Karte sichern wollen, in dem man mit „**dd**“ die Daten der SD-Karte in eine Imagedatei schreibt, würde der gesamte Speicherinhalt von 8 GB kopiert werden, womit auch die Imagedatei diese Größe erhalten würde. Hingegen ermittelt **iread.sh** die Obergrenze der höchsten Partition auf der SD-Karte und kopiert in diesem Fall lediglich 3 GB in die Imagedatei, die schlussendlich auch nur diese Größe annimmt.

Auch wenn **iread.sh** nur lesend auf Gerätedateien zugreift, womit der Schaden einer versehentlichen Fehleingabe bei weitem nicht so fatal wäre wie bei **iwrite.sh**, wirkt der gleiche Sicherheitsmechanismus wie bei **iwrite.sh**, indem eine die Quell-Gerätedatei mit den Einträgen der Datei `/etc/dev_permissions.conf` abgeglichen wird.

Was macht iread?

- Zuerst werden mögliche Optionen ausgewertet.
- Es werden die weiteren Aufrufparameter untersucht. Wenn **iread** nicht im Dialogbetrieb aufgerufen wird (Option -d oder -x sind nicht gesetzt) und keine weiteren Parameter vorhanden sind, wird das Programm mit einer Fehlermeldung beendet.
- Falls **iread** im Dialogbetrieb aufgerufen wird, (mit Option -d oder -x) und keine weiteren Parameter vorhanden sind, Erscheint ein Dateiauswahlmenü, um den Namen der Ziel-Imagedatei einzugeben oder eine vorhandene auszuwählen damit diese überschrieben wird. Das Dateiauswahlmenü erscheint auch, wenn als erster Parameter statt einen voll qualifizierten Dateinamen (/Pfad/Dateiname) nur der Pfad angegeben wird. In diesem Fall Zeigt das Dateiauswahlmenü die Dateien im angegebenen Pfad.
- Falls der zweite Parameter fehlt, sucht **iread** bei allen in der Datei `/etc/dev_permissions.conf` angegebenen Gerätedateien nach einem passenden Speichermedium. Wird mehr als ein Speichermedium gefunden, erscheint im Dialogbetrieb (Optionen „-d“ oder „-x“) eine Auswahldialogbox mit den Gerätedateien und der Speicherkapazität der damit verknüpften Speichermedien,

3 Partitionen in einer Imagedatei kommen auf dem Arbeitsrechner auch mit **iparted.sh** bearbeitet werden.

welche als Ziel in Frage kommen. Andernfalls wird das Programm einer entsprechenden Fehlermeldung abgebrochen.

- Falls der zweite Parameter angegeben wird, der aus dem voll qualifizierten Namen der Gerätedatei, welche mit dem Quelldatenträger verknüpft ist besteht, oder nur aus dem letzten Buchstaben z.B. „/dev/sde“ oder nur „e“ (für MS-Windows-Nostalgiker), wird diese Eingabe mit der Datei `/etc/dev_permissions.conf` verglichen, falls in dieser Datei kein entsprechender Eintrag gefunden wird, bricht das Programm seine Ausführung mit einer entsprechenden Fehlermeldung ab. Sollte der zweite Parameter aus einem symbolischen Link auf eine Gerätedatei bestehen wird dieser vor dem Vergleich von **iread** dereferenziert.
- Es wird untersucht, ob die Ziel-Imagedatei bereits vorhanden ist. Wenn das der Fall ist, stellt **iread** eine entsprechende Rückfrage und fragt, ob diese Datei überschrieben werden darf. Wenn diese Frage negativ beantwortet wird, wird die Programmausführung beendet.
- Fells die Option -e, --entire nicht gesetzt wurde, versucht **iread** die effektive Anzahl der zu lesenden Bytes anhand der Obergrenze der höchsten auf dem Datenträger gefundenen Partition zu ermitteln. Falls keine Partitionen auf den Datenträger gefunden wurden und die Option -Y, --yes nicht gesetzt wurde, stellt **iread** eine Rückfrage,und fragt, ob es die Programmausführung fortsetzen soll. Bei einer positiven Antwort wird als effektive Anzahl der zu lesenden Bytes die komplette Speicherkapazität des Quell-Datenträgers angenommen. Das hätte die gleiche Wirkung als wäre die Option -e, --entire gesetzt.
- Falls die Option -Y, --yes nicht gesetzt wurde, erscheint nun eine Rückfrage, bei der nochmal der Quell-Datenträger und der Name der Zieldatei angezeigt werden. Wird die Frage negativ quittiert, wird die Programmausführung beendet.
- Nun untersucht **iread** ob der Quell-Datenträger irgendwo im System eingehängt (gemounted) ist. Ist das der Fall, versucht **iread** diesen auszuhängen (umount). Falls diese Aktion fehlschlägt, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Ein Fehlschlag könnte daran liegen, dass irgend ein oder mehrere Programme (z.B. der Dateimanager) auf den Quell-Datenträger zugreift.
- Als nächstes untersucht **iread** nochmal ob die angegebene Zieldatei bereits existiert. Ist das der Fall untersucht **iread**, ob die vorhandene Zieldatei irgendwo im System eingehängt (gemounted) ist. Falls das zutrifft, versucht **iread** mittels dem Programmaufruf **imount.sh -u** die Zieldatei auszuhängen, wobei sich **iread** den alten Einhängepunkt merkt um später die frisch eingelesene Imagedatei an dieser Stelle wieder einzuhängen. Falls das Aushängen nicht gelingen sollte, was daran liegen könnte, dass irgend ein oder mehrere Programme (z.B. Der Dateimanager) noch auf die Zieldatei zugreifen, wird das Programm mit einer entsprechenden Fehlermeldung beendet.
- Nun beginnt der Lesevorgang. Falls die Option -s, --silence nicht gesetzt wurde, wird der Lesefortschritt über einen Fortschrittsbalken angezeigt. Falls **iread** nicht im Dialogbox-Modus läuft, (Option -d, --dialog oder -x, --xdialog sind nicht gesetzt) wird der Kopierfortschritt mittels „=“-Zeichen angezeigt, sonst in der entsprechenden Dialogbox.
- Nach erfolgreichen Abschluss des Lese-Kopiervorgangs, versucht **iread** die frisch

eingelesene Imagedatei wieder an dem alten Einhängepunkt, mittels dem Programmaufruf **imount.sh** einzuhängen, falls **iread** vorher eine ältere Imagedatei mit dem gleichen Namen ausgehängt hatte.

Aufruf von iread

Aufruf:

```
iread.sh [options] [Target-Imagefile] [Source Block-Device]
```

Option -h, --help

Anzeige der Hilfe.

Option -v

Ausgabe von zusätzlichen Informationen auf der Textkonsole.

Option -s, --silence

Es wird kein Fortschrittsbalken angezeigt nach der Sicherheitsüberprüfung wird für den Kopiervorgang lediglich **dd** aufgerufen.

Option -d, --dialog

Dialogbetrieb in pseudografischen Dialogboxen.

Option -x, --xdialog

X-Windows Dialogbetrieb.
Falls der Programmaufruf im Usermodus via **sudo** nicht funktioniert, sollte statt dessen **xsudo.sh** verwendet werden, oder man muss sich vorher via **su** als root anmelden.

Option -l

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf` angezeigt.

Option -c

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf` angezeigt, die zur Zeit mit einem physikalischen Speichermedium verknüpft sind.

Option -e, --entire

Wenn diese Option gesetzt wurde, wird das komplette Datenvolumen des externen Quell-Datenträgers in die Zieldatei kopiert, womit die Zieldatei auch die gleich Größe erhält wie die des Quell-Datenträgers. Ansonsten werden nur die Daten bis zur Obergrenze der höchsten gefundenen Partition kopiert, falls welche gefunden wurden.

Diese Option ist nur sinnvoll, wenn sich noch weitere Daten auf den Quelldatenträger

befinden befinden sollten, die oberhalb der höchsten Partition liegen.

Änderungen, der in dem Quell-Datenträger befindlichen Partitionen, können mit dem Programm **iperted.sh** gemacht werden.

Option -n

Es findet findet kein Kopiervorgang statt. Diese Option dient für Testzwecke und der Programmwartung.

Option -Y, --yes

Diverse Sicherheitsabfragen werden übersprungen. bzw. automatisch positiv quittiert. Diese Option ist nur wirksam, wenn das externe Quell-Speichermedium beim Programmaufruf explizit im zweiten Parameter angegeben wurde.

Option -L=<RATE>

Limitiert die Übertragungsrate für das Programm „**pv**“ welches – wenn Option „-s“ nicht gesetzt wurde – den Kopiervorgang ausführt, in Bytes pro Sekunde. Die Einheit "k" für Kilobyte , "m" für Megabyte, "g" für Gigabyte, oder "t" für Terabyte kann hinter die Zahl gesetzt werden. Der Standardwert ist auf 5m, also 5 Megabyte pro Sekunde, eingestellt. Falls die Anzeige vom Fortschrittsbalken stark von tatsächlichen Kopierfortschritt differiert, z.B. wenn der Fortschrittsbalken schon bei 100% steht, aber der eigentliche Kopiervorgang erst nach einer deutlichen Verzögerung abgeschlossen ist, macht es Sinn eventuell mit dieser Option einen anderen Wert auszuprobieren.

Beispiele

Die Beispiele sind adäquat zu den Beispielen von **iwrite**.

imount (Image-Mounter)

Mit diesem Programm können Dateisysteme auf den Partitionen einer Imagedatei oder eines externen Wechseldatenträgers (Block-Device) an einem beliebigen Verzeichnisstelle im Arbeits-PC eingehängt (gemountet) oder ausgehängt werden. Dabei kann es sich um alle einhängbaren Partitionen in der Imagedatei oder dem Datenträger handeln oder gezielt um einzelne Partitionen. Bei einem externen Datenträger reicht es auch den letzten Buchstaben anzugeben, also statt „/dev/sde“ nur „e“. Beim Einhängen einer oder mehrerer Partitionen erzeugt **imount** für jede einzelne Partition ein Unterverzeichnis an dem das Dateisystem der betreffenden Partition eingehängt wird. Dieses Unterverzeichnis wird aus dem Namen und Verzeichnispfad der Imagedatei oder Gerätedatei und der Partitionsnummer erzeugt.

Da eine eindeutige Datei (voll qualifizierter Dateiname) mit ihren Dateinamen *und* ihrem Verzeichnispfad eindeutig identifiziert werden kann, muss bei der Generierung eines Einhängepunktes der Verzeichnispfad im Namen enthalten sein. hierbei werden die Pfad-Trenner (Pfad-Separatoren) „/“ durch „+“ ersetzt[4].

Zum Beispiel wird aus:

`/Pfad/zu/der/Datei/pi.img`

Der Einhängepunkt für Partition 1:

`/mnt/+Pfad+zu+der+Datei+pi.img1/`

und der Einhängepunkt für Partition 2:

`/mnt/+Pfad+zu+der+Datei+pi.img2/`

Sofern die Datei `pi.img` im Verzeichnis `/Pfad/zu/der/Datei/` mindestens 2 einhängbare Partitionen enthält (Partition 1 und Partition 2).

Bein Aushängen einer oder aller Partitionen (Option -U), wird das jeweilige betreffende Unterverzeichnis ebenfalls wieder gelöscht.

Wenn für **imount** kein zentraler Einhängepunkt (Mountpoint) angegeben wird, wird als Standart-Einhängepunkt **`/mnt/`** verwendet.

Wird zum Einhängen ein externer Datenträger - also eine Gerätedatei - angegeben, wird diese vorher mit der Datei **`dev_permissions.conf`** abgeglichen. Falls die Gerätedatei dort nicht gefunden wird, wird **imount** mit einer entsprechenden Fehlermeldung beendet.

Was macht imount?

- Zuerst werden mögliche Optionen ausgewertet. Bei der Option -U oder -u werden die Partitionen einer Imagedatei oder eines externen Datenträgers wieder ausgehängt.
- Es werden die Parameter geprüft. Handelt es sich bei dem ersten Parameter um einen externen Datenträger – also einer Gerätedatei (Block-device) – wird der

4 Es sollte daher vermieden werden das „+“- Zeichen in einem Dateinamen für eine Imagedatei zu verwenden.

Name – falls es ein symbolischer Link sein sollte – dereferenziert und mit der Datei **`/etc/dev_permissions.conf`** abgeglichen. Wird der Dateiname dort nicht gefunden, bricht **imount** seine Programmausführung mit einer entsprechenden Fehlermeldung ab.

- Es wird untersucht, ob noch ein weiterer Aufrufparameter vorhanden ist. Falls kein weiterer Parameter vorhanden ist, wird als zentraler Einhängepunkt, für alle einhängbaren Partitionen aus dem ersten Parameter, das Verzeichnis **/mnt/** verwendet. Andernfalls wird die Zeichenkette im zweiten Parameter als zentraler Einhängepunkt verwendet. Ist dieser nicht vorhanden, wird er von **imount** angelegt.
- Es wird untersucht, ob die angegebene Imagedatei oder externe Datenträger bereits eingehängt wurde. Falls ja, wird eine Warnung ausgegeben und gefragt, ob die Programmausführung trotzdem fortgesetzt werden soll.
- Es wird untersucht, ob die Imagedatei oder externe Datenträger einhängbare Partitionen enthält. Falls keine Partitionen gefunden werden und es sich dabei um einen externen Datenträger handelt, wird dieser als Partition „0“ gehandelt, d.h. es wird ein Einhängepunkt angelegt, der mit der Ziffer „0“ endet. Andernfalls werden im zentralen Einhängepunkt für alle einhängbaren Partitionen entsprechende Unterverzeichnisse erstellt, die sich aus dem Pfad zur Imagedatei und dem Dateinamen zusammensetzen und jeweils mit der Nummer der dort eingehängten Partition enden.
- Es wird für jede gefundene einhängbare Partition der Sektor-Offset ermittelt und im Bytes umgerechnet, welcher als Offset für die Funktion „mount“ verendet wird. Alle so gefundenen partitionen werden im zentralen Einhängepunkt in ihren jehweiligen zugehörigen Unterverzeichnis eingehängt (gemountet), falls mit der Option „-P“ nicht explizit Partitionsnummern angegeben wurden.

Aufruf von imount

Auftuf:

```
imount [optionen] <Imagedatei oder Gerätedatei> [Zentraler
                    Einhängepunkt]
```

Option -h, --help

Anzeigen der Hilfe.

Option -v

Anzeigen von zusätzlichen Informationen während der Programmausführung. (Verbose)

Option -u

Aushängen der Partitionen ohne dass dabei der jeweilige Einhängepunkt (Mountpoint) gelöscht wird. Falls zusätzlich der Parameter -P angegeben wurde, werden nur die dort angegebenen Partitionen ausgehängt.

Beispiel:

```
~> sudo imount –u /Pfad/zu/der/Datei/pi.img
```

Option -U

Wie Option -u, jedoch werden hier die entsprechenden Einhängepunkte wieder gelöscht, falls das möglich ist.

Beispiel:

```
~> sudo imount –U /Pfad/zu/der/Datei/pi.img
```

Option -n

Diese Option ist für Testzwecke. Das ein oder aushängen wird nur simuliert.

Option -m

Für eine gefundene Boot-Partition wird der Einhängepunkt am Schluss mit einem „B“ markiert.

Option -i

Liefert die Einhängepunkte der im ersten Parameter angegebenen Imagedatei oder externen Datenträger zurück, falls dieser eingehängt wurde. Andernfalls nichts.

Beispiel:

```
~> sudo imount –i /Pfad/zu/der/Datei/pi.img
```

Ergebnis:

```
/mnt/+Pfad+zu+der+Datei+pi.img1
/mnt/+Pfad+zu+der+Datei+pi.img2
```

Option -l

Listet die erlaubten Gerätedateien aus der Datei **`/etc/dev_permissions.conf`** auf.

Option -L

Listet die Partitionen der angegebenen Imagedatei oder eines externen Datenträgers auf. Falls dieser mittels **imount** eingehängt wurde, wird in der letzten Spalte der zur jeweiligen Partition zugehörige Einhängepunkt (Mountpoint) angezeigt.

Beispiel:

```
~> imount –vL /dev/sde
```

Führt zu dem Ergebnis:

```
PartNr.          id              mounted
1                c               –
2                83              /mnt/+dev+sde2
```

In diesem Beispiel wurde eine zuvor lediglich die Partition 2 einer SD-Karte mittels dem Befehl:

```
~> sudo imount -P=2 /dev/sde
```

eingehängt. Ohne die zusätzliche Option „v“ würde die Spaltenbezeichnung in der ersten Zeile fehlen.

Option -E

Zusätzliche Option für -L. Mit der Kombination -LE werden noch erweiterte Partitionen angezeigt, falls sie auf dem Datenträger oder Imagedatei vorhanden sind.

Option -S

Zusätzliche Option für -L. Mit der Kombination -LS wir eine eventuell vorhandene Swap-Partition des Datenträgers oder der Imagedatei angezeigt.

Option -N

Wenn diese Option gesetzt wurde, wird bei der Namensgenerierung der Einhängepunkte lediglich der Dateiname der Imagedatei oder Gerätedatei des externen Datenträgers plus abschließender Partitionsnummer verwendet und nicht die zusätzliche Pfadangabe als Prefix. **ACHTUNG: Eine mit diesem Parameter eingehängte Imagedatei oder Gerätedatei muss auch mit diesem Parameter wieder ausgehängt werden. Also mit der Kombination -uN oder -UN.**

Option -s

Liefert das Ersatzzeichen für den Pfadtrenner „/“ (Separator) zurück. Also „+“.

Option -p

Gibt den zentralen Einhängepunkt (Base-Mountpoint) einer eingehängten Imagedatei oder Gerätedatei zurück, andernfalls nichts.

Option -P

Mit dieser Option können einzelne Partitionen einer Imagedatei oder eines externen Datenträgers eingehängt oder in Kombination der Option -u oder -U ausgehängt werden.

Als Parameter erwartet diese Option hinter dem „=“-Zeichen eine oder mehrere durch Komma getrennte Partitionsnummern, welche mittels der Option -L in Erfahrung gebracht werden können.

Beispiel 1:

Einhängen der Partitionen 1 und 5 einer Imagedatei:

sudo imount –P=1,5 /Pfad/zu/der/Datei/ArchLin.img

Beispiel 2:

Aushängen der Partition 1 der selben Imagedatei.

sudo imount –UP=1 /Pfad/zu/der/Datei/ArchLin.img

Wird diese Option weggelassen, tut **imount** grundsätzlich alle einhängbaren Partitionen einhängen, bzw. mittels den Optionen -u oder -U wieder aushängen.

Beispiele

1.)

Einhängen einer Imagedatei `/home/donald/images/pi.img` in den zentralen Einhängepunkt „`/home/donald/wurzel`“:

sudo imount /home/donald/images/pi.img /home/donald/wurzel

Aushängen derselben Imagedatei mit löchung der zuvor generierten Verzeichnisse:

sudo imount –U /home/donald/images/pi.img /home/donald/wurzel

2.)

Einhängen an den Standard-Einhängepunkt **/dev/mnt/** eines externen Datenträgers der mit der Gerätedatei **„/dev/sdc/“** verknüpft ist, wobei hier die verkürzte Schreibweise verwendet wird:

sudo imount /dev/sdc

Alternativ (für MS-Windows-Nostalgiker):

sudo imount c

Aushängen desselben externen Datenträgers und Entfernung der Einhängepunkte

sudo imount –U /dev/sdc

Alternativ (für MS-Windows-Nostalgiker):

sudo imount –U c

cross-chroot (Chroot-Umgebung und Emulation von Fremdprozessoren)

Mit dem „cross-chroot“ kann auf einem Linux-x86 Pcvon einer Linux-Image-Datei direkt eine virtuelle Linux-Umgebung geschaffen werden, die auch auf Fremdprozessoren z.B. ARM basieren. Somit wird es möglich Programme welche z.B. für einen ARM-Prozessor compiliert sind auf dem x86-PC zum laufen zu bringen[5].

In anderen Worten, es wird somit möglich z.B. Programme für den Raspberry Pi, Beagle Bone, etc. auf dem PC zu starten und auszuführen. Als Parameter benötigt cross-chroot lediglich den Namen der entsprechenden Imagedatei oder den Pfad zu einem virtuellen Root-Verzeichnis, auf dem der Inhalt einer Root-Partition kopiert wurde, oder eines externen Datenträgers auf dem ein entsprechendes Linux-Image (z.B. mittels **iwrite**) geschrieben wurde.

Linux-Imagedateien können somit direkt auf dem PC weitgehend getestet werden ohne die dafür entsprechende Hardware, z.B. den Raspberry Pi oder Beagle-Bone, zu benötigen, was u.U. viel Zeit ersparen kann.

Bei Linux-Imagedateien, die auf einer Debian-Distribution basieren, ist es möglich direkt auf der Imagedatei oder auf einem virtuellen Root-Verzeichnis, worauf der Inhalt einer Root-Partition kopiert wurde, „**apt-get** Funktionen“ aufzurufen. Somit ist es möglich, direkt auf der Imagedatei Programme zu installieren[6] oder zu deinstallieren und zu updaten. Damit kann eine Imagedatei oder ein virtuelles Root-Verzeichnis, auf dem aktuellsten Stand gehalten werden. Besonders nützlich ist es damit diverse Bibliotheken (z.B. QT- und KDE- Librarys) zu installieren und auf dem aktuellsten Stand zu halten. Das ist zum Beispiel sinnvoll, wenn ein Cross-Compiler (z.B. arm-linux-gnueabi-gcc) auf Bibliotheken der Imagedatei oder auf einem virtuellen Root-Verzeichnis zugreift.

Bei Wheezy Imagedateien für den Rasberry Pi funktionieren auch die meisten Menüpunkte von dem Konfigurationsprogramm **„raspi-config“**[7]. Der Menüpunkt 1 „Expand Filesystem“ kann in diesem Fall nicht funktionieren, hierzu kann aber das I-Tool-Programm **iparted** verwendet werden.

Falls **cross-chroot** auf einen externen Datenträger oder auf eine Imagedatei angewendet wird, interpretiert bei der Option **-f cross-chroot** die im Image befindliche Dateisystemtabelle **„/etc/fstab“** und versucht die dort enthaltenen Einträge entsprechend zu mounten, sofern die entsprechenden Komponenten gefunden werden und die Datei **fstab** nicht all zu exotisch ist.

Das Motto von cross-chroot ist: Eine virtuelle Linux-Umgebung wird wieder so verlassen wie sie vorgefunden wurde. Das heißt, alle Einhängepunkte und temporären Installationen, die für den Betrieb einer virtuellen Linux-Umgebung notwendig sind und von **corss-chroot** durchgeführt wurden, werden beim ordnungsgemäßen Beenden von **cross-chroot** wieder von diesem rückgängig gemacht.

---

5 Das gilt allerdings nur für Programme, die nicht auf spezielle Hardware zugreifen.

6 Natürlich muss die Imagedatei und die darin enthaltenen Partitionen für nachträgliche Programm- und Bibliotheksinstallationen groß genug sein, was in der Regel, direkt nach dem Download vom Internet nicht der Fall ist. Mit dem Programm **iparted** kann die Dateigröße einer Imagedatei, zusammen mit den darin enthaltenen Partitionen, verändert werden.

7 Bei raspi-config sollte cross-chroot mit den Optionen -m und -f aufgerufen werden.

Was macht cross-chroot?

Cross-chroot ist die Zusammenfassung von 3 Programmen:

1. Der Image-Mounter **imount**, welcher jedoch nur bei Imagedateien oder externen Datenträgern zum Einsatz kommt.
2. Das Programm zum verändern des Root-Verzeichnisses im aktuellen Prozess **chroot** (change root).
3. Die virtuelle Maschine **qemu** (Quick Emulator), welche einen Fremd-Prozessor (z.B. ARM) auf dem Host-Prozessor (Arbeits- PC) emuliert. Dabei ermittelt **cross-chroot** automatisch anhand der ausführbaren Binärdateien im vorliegenden Image, um welchen Prozessor es sich dabei handelt und installiert darauf die entsprechende **qemu**- Variante auf des Image. Beim Beenden von **cross-chroot**, wird diese Installation wieder rückgängig gemacht.

Als obligatorischen Parameter benötigt **cross-chroot** entweder

1. den voll qualifizierten Namen einer Imagedatei oder
2. den voll qualifizierten Namen oder letzten Buchstaben („/dev/sde“ oder nur „e“) einer Gerätedatei, welche mit einem externen Datenträger verknüpft ist oder
3. den Pfad zu einer virtuellen Root-Umgebung in die die Root-Partition eines Linux-Images kopiert wurde.

Als weitere optionale Parameter kann direkt ein Programm mit samt seinen Optionen und Parametern angegeben werden, welches in der virtuellen Linux-Umgebung aufgerufen werden soll. Fehlt dieser Parameter, wird statt dessen automatisch die Kommandoshell in der virtuellen Linux-Umgebung aufgerufen, welche mit der Eingabe von **„exit“** verlassen werden kann und womit auch **cross-chroot** ordnungsgemäß beendet wird.

Der (grobe) Ablauf:

- Zuerst werden mögliche Optionen ausgewertet.
- Es wird geprüft, ob bereits eine Instanz von **cross-chroot** mit dem selben Image gestartet wurde. Trifft das zu, stellt cross-chroot die frage, ob es die bereits laufende Instanz von sich selbst terminieren soll. Wenn sie Frage mit „y“ quittiert wird, versucht cross-chroot die bereits gestartete Instanz von sich zu beenden. Danach wird auch die aktuelle Instanz auf jeden Fall beendet, womit **cross-chroot** erneut aufgerufen werden soll.
- Der obligatorische Parameter wird untersucht. Handelt es sich dabei um ein Verzeichnis, wird untersucht, ob es sich bei diesem Verzeichnis auch um eine virtuelle Linux-Root-Umgebung handelt. trifft das nicht zu, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Falls es sich bei dem Parameter um eine Gerätedatei oder einen symbolischen Link auf eine solche handelt, wird der Parameter mit den Einträgen in der Datei **`/dev/dev_permissions.conf`** abgeglichen, wird dort kein entsprechender Eintrag gefunden, wird das Programm mit einer entsprechenden Fehlermeldung beendet.
- Falls es sich bei dem obligatorischen Parameter um eine Gerätedatei auf einen externen Datenträger oder eine Imagedatei handelt, wird nun untersucht, ob

diese Datei via dem Programm **imount** bereits eingehängt wurde. Ist das nicht der Fall, werden jetzt alle in der Datei gefundenen Partitionen via **imount** eingehängt (gemounted) und sich dieser Vorgang gemerkt, damit bei der Beendigung von cross-chroot dieser Vorgang wieder rückgängig gemacht wird (aushängen, unmounten).

- Bei einer Gerätedatei (Block-Device) oder Imagefile wird nun bei allen eingehängten (gemounteten Partitionen) nach der Root-Partiton gesucht. Wird keine gefunden, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Wurde eine Root-Partition gefunden und die Option **-f** ist gesetzt, wird nun die in der Root-Partition befindliche Dateisystemtabelle z.B. **`/mnt/+Pfad+zum+Mountpoint+pi.img2/etc/fstab`** gelesen und versucht die dort befindlichen Einträge zu mounten, sofern die jeweiligen erforderlichen Komponenten gefunden werden und die Datei **fstab** nicht all zu exotisch ist.
- Anhand der in der Root-Partition gefundenen ausführbaren Binärdateien wird an deren Muster nun ermittelt, für welche CPU diese compiliert worden sind. Wird eine für **cross-chroot** (noch) unbekannte CPU ermittelt[8], werden alle bereits von diesem Programm durchgeführten Aktionen rückgängig gemacht und das Programm mit einer entsprechenden Fehlermeldung beendet.
- Anhand des ermittelten CPU-Typs wird nun geprüft, ob hierfür eine entsprechende Variante von **qemu** vorhanden ist. Wird keine diesbezügliche Emulatorvariante gefunden, beendet sich das Programm, unter Rücknahme aller bereits erfolgten Aktionen, mit einer entsprechenden Fehlermeldung.
- Der ermittelte Emulatorvariante von **qemu** wird nun in das Prozess-Dateisystem **`„/proc/sys/fs/binfmt_misc/register"`** eingetragen, falls dies nicht schon bei einem früheren Aufruf von **cross-chroot** oder einem anderen Programm erfolgte. Diese Maßnahme veranlasst das Betriebssystem, bei jedem Aufruf einer ausführbaren Binärdatei, diese mit einen dort hinterlegten Bit-Muster zu vergleichen, welches für den jeweiligen CPU-Typ typisch ist. Liegt eine Musterübereinstimmung vor, wird diese Binärdatei nicht direkt aufgerufen, sondern eine mit diesem Bit-Muster verknüpfter Emulator, der als Argument diese Binärdatei bekommt. Falls diese Aktion fehlschlägt beendet sich das Programm, unter der Rücknahme aller vorhergehenden Aktionen, mit einer entsprechenden Fehlermeldung.
- Falls die Option **-m** gesetzt wurde, werden nun einige systemrelevante Verzeichnisse und Gerätedateien des Betriebssystems in Root-Partion eingehängt. Falls die Option **-m** ohne weitere Parameter angegeben wird, ist das : „/dev“, „/dev/pts“, „/proc“ und „/sys“. ACHTUNG: Dies beinhaltet auch ein Sicherheitsrisiko!
- Nun wird die ermittelte Emulatorvariante von **qemu** in die eingehängte Root-Partition kopiert. z.B.: von **`/usr/local/bin/qemu-arm`** zu **`/mnt/+Pfad+zum+Mountpoint+pi.img2/usr/local/bin`**. Diese Maßnahme ist notwendig, da nach dem bevorstehenden Aufruf von **chroot** der aktuelle Prozess unterhalb von dem Verzeichnis mit der virtuellen Root-Partition

8 Momentan kennt **cross-chroot** nur i386 und ARM, jedoch ist das Programm leicht auf weitere CPU-Typen ausbaubar.

nichts mehr „sehen“ kann. Bei der Beendigung von **cross-chroot**, wird die in der virtuellen Root-Partition hinein kopierte Variante wieder gelöscht.

- Falls mit der Option „**--enter=**“ ein Programm übergeben wurde, wird nun dieses gestartet. Als Parameter bekommt dieses Programm den Verzeichnispfad in der sich die virtuelle Root-Partition befindet, also den selben Parameter mit der danach **chroot** aufgerufen wird. Gibt dieses Programm einen Wert zurück der von Null verschieden ist, wird die Programmausführung unter Rücknahme aller vorhergehenden Aktionen beendet. Der Rückgabewert von cross-chroot ist in diesem Fall der Rückgabewert von dem unter „--enter“ angegebenen Programm.
- Jetzt wird **chroot** gestartet. Als ersten Parameter bekommt **chroot** den Verzeichnispfad in der sich die virtuelle Root-Partition befindet. Die weiteren Parameter (falls noch vorhanden) sind die Parameter die **cross-chroot** als optionale Parameter übergeben wurden. Sind keine weiteren Parameter vorhanden, startet **chroot** eine Eingabekonsole (in der Regel **`/bin/bash`**). In diesem Fall kann **chroot**, und damit auch **cross-chroot**, mir der Eingabe von **„exit“** beendet werden.
- Nach der Beendigung von **chroot** wird nun geprüft, ob mit der Option „**--leave=**“ ein Programm übergeben wurde. Ist das der Fall, wird dieses nun gestartet. Als Parameter bekommt dieses Programm den Verzeichnispfad in der sich die virtuelle Root-Partition befindet, also den selben Parameter mit der zuvor **chroot** aufgerufen wurde. Als weiteren Parameter den Rückgabewert von **chroot**, welcher zugleich der Rückgabewert des letzten Programms in der Chroot-Umgebung war. Falls der Rückgabewert des unter der Option “**--leave**“ angegebenen Programms von Null verschieden ist, wird dieser auch als Rückgabewert von **cross-chroot** verwendet.
- Alle Aktionen, die **cross-chroot** vor dem Aufruf von **chroot** auf der virtuellen Root-Partition gemacht hatte, werden nun in der umgekehrten Reihenfolge rückgängig gemacht. Sollte **cross-chroot** im Fall eines externen Datenträgers oder einer Imagedatei diesen selber via **imount** eingehängt (gemountet) haben, werden nun alle eingehängten Partitionen mittels dem programminternen Aufruf von **imount -U** wieder ausgehängt.
- Falls **cross-chroot** ordnungsgemäß beendet wurde, entspricht der Rückgabewert von **cross-chroot** dem Rückgabewert des letzten in der Chroot-Umgebung gestarteten Programms.

Aufruf von cross-chroot

Aufruf:

```
cross-chroot [OPTION] <target-root> [COMMAND [ARG]...]
```

Option -h, --help

Anzeigen der Hilfe.

Option -v

Ausgabe zusätzlicher Informationen auf der Eingabeshell (verbose).

Option -l

Listet die erlaubten Gerätedateien aus der Datei **`/etc/dev_permissions.conf`** auf.

Option -f

Wenn diese Option angegeben wird, versucht **cross-chroot** die in der virtuellen Root-Partition befindliche Dateisystemtabelle „fstab“
(z.B. **`/mnt/+Pfad+zum+Mountpoint+pi.img2/etc/fstab`**)
zu interpretieren. Diese Option ist nur wirksam, wenn als obligatorischer Parameter <target-root> eine Gerätedatei, die mit einem externen Datenträger verknüpft ist, oder eine Imagedatei angegeben wird. Im Falle eines Verzeichnisses, wird eine entsprechende Warnung ausgegeben. Falls die Datei **fstab** nicht all zu exotisch ist, versucht cross-chroot alle in **fstab** befindlichen Einträge einzuhängen (zu mounten). Bei den gängigen Linux-Images für embedded Devices ist das in der Regel die Boot-Partition die in /boot/ eingehängt wird. Z.B. **`/mnt/+Pfad+zum+Mountpoint+pi.img1`** wird in **`/mnt/+Pfad+zum+Mountpoint+pi.img2/boot`** eingehängt [9].
Daher ist diese Option sinnvoll, wenn z.B. innerhalb der Chroot-Umgebung eines Debian-Images eine Paketaktualisierung via dem Funktionsaufruf **`apt-get upgrade`** gemacht wird. In diesen Fall könnte z.B. auch der Linux-Kern „**kernel.img**“ ersetzt werden. Dieser befindet sich jedoch nicht auf der Root-Partition, sondern auf der Boot-Partition. Jedoch das Programm **apt-get** „sieht“ den Linux-Kern, und andere Dateien wie z.B. **start.elf**, in dem in der Datei **fstab** angegebenen Mountpoint (meistens **`/boot/`**).

Option -m

Mit Angabe dieser Option werden Systemressourcen vom Host- Betriebssystem (also vom PC) in die cross-chroot-Umgebung eingehängt (gemounted). Bei ordnungsgemäßer Bendigung von cross-chroot werde alle hierdurch eingehängten Ressourcen wieder ausgehängt (geumounted).

ACHTUNG: Hiermit ist der ursprüngliche Sicherheitsaspekt, für das angeblich das

9 Im Fall eines Images für den BeagleBone wäre der Mountpoint für die Boot-Partition z.B. **`/mnt/+Pfad+zum+Mountpoint+beagle.img2/boot/uboot`** daher muss auf jeden Fall fstab ausgewertet werden.

Programm **chroot** geschaffen wurde nicht mehr gegeben, da nun auch Programme der Chroot-Umgebung, die auch Schadprogramme sein könnten, Zugriff auf die Systemressourcen das PCs haben!

Daher sollte diese Option mit bedacht gewählt werden. Sollte allerdings ein Upgrade z.B. via **apt-get** gemacht werden, wird diese Option unumgänglich sein, da **apt-get** Zugriff auf das Internet braucht.

Wenn Option **-m** ohne weitere Parameter (mit = oder +) gesetzt wird, hängt **cross-chroot** folgende Systemverzeichnisse an die adäquate Stelle der Chroot-Umgebug – also der virtuellen Root-Partition – ein: **`/dev`**, **`/dev/pts`**, **`/proc`** und **`/sys`**.

Alternativ hierzu kann mit der Option -m eine explizite Liste von Ressourcen mit dem „=“ Zeichen angegeben werden. Die einzelnen Elemente einer solchen Liste werden jeweils mit einem Doppelpunkt „:“ getrennt.

Beispiel:

```
~> sudo cross-chroot -m=/dev:/proc:/sys pi.img
```

Alternativ zu der expliziten Angabe der Ressourcenliste mittels „=“ kann mittels „+“ auch au der Standardressoucenliste auch ergänzende Ressourcen angefügt werden.

Beispiel:

```
~> sudo cross-chroot -m+/tmp pi.img
```

Option -p=

Mit dieser Option kann explizit ein Basis-Einhängepunkt (Mountpoint) angegeben werden, falls **cross-chroo**t als obligatorischen Parameter eine Imagedatei oder eine Gerätetatei, die mit einem externen Datenträger verknüpft ist, bekommt. Wenn diese Option nicht angegeben wird, wird der Standard-Einhängepunkt „**`/mnt`**“ verwendet.

Beispiel:

```
~> sudo cross-chroot -p=/Pfad/zum/Mountpoint pi.img
```

Option --enter=

Mit dieser Option kann ein zusätzliches Programm bzw. Shellscript angegeben werden, welches von **cross-chroot** unmittelbar vor dem Wechsel in die Chroot-Umgebung – also unmittelbar vor dem Programmaufruf von **chroot** – aufgerufen wird. Zu diesem Zeitpunkt sind bereits alle anderen Vorbereitungen für den Eintritt Chroot-Umgebung getätigt worden. Als Aufrufparameter erhält das betreffende Programm den Verzeichnispfad zur virtuellen Root-Partition. Hiermit können noch zusätzliche vorbereitende Maßnahmen getätigt werden, welche nicht von **cross-chroot** erledigt werden.

Wenn dieses Zusatzprogramm einen Rückgabewert hat, der ungleich Null ist, wird cross-chroot vorzeitig, unter Rücknahme aller vorher getätigten Aktionen auf der virtuellen Root-Partition, beendet. Der Rückgabewert von cross-chroot entspricht in diesen Fall dem des Zusatzprogramms.

Beispiel 1:

```
~> sudo cross-chroot --enter=Zusatzprogramm pi.img
```

Beispiel 2:

```
~> sudo cross-chroot --enter="Zusatzprogramm mitParameter" pi.img
```

Falls das Zusatzprogramm noch weitere Parameter wie im Beispiel 2 benötigt, muss die komplette Signatur in Anführungszeichen gesetzt werden. Der Pfad zu der virtuellen Root-Partition wird von **cross-chroot** in diesem Fall als letzter Perameter angehängt.

Option --leave=

Mit dieser Option kann ein zusätzliches Programm bzw. Shellscript angegeben werden, welches von **cross-chroot** unmittelbar nach Beendigung der virtuellen Chroot-Unhebung – also nach Beendigung von dem Programm **chroot** – aufgerufen wird.

Zu diesem Zeitpunkt sind noch alle Aktionen, die **cross-chroot** vor dem Eintritt Chroot-Umgebung auf der virtuellen Root-Partition getätigt hatte gültig.

Als Aufrufparameter erhält das betreffende Programm den Verzeichnispfad zur virtuellen Root-Partition und unmittelbar als weiteren Parameter den Rückgabewert von chroot, welcher dem Rückgabewert von dem zuletzt aufgerufenen Programm in der Chroot-Umgebung ebtspricht.

Wenn das Programm einen Rückgabewert hat, der ungleich Null ist, wird dieser auch als Rückgabewert von cross-chroot verwendet.

Beispiel 1:

```
~> sudo cross-chroot --leave=Zusatzprogramm pi.img
```

Beispiel 2:

```
~> sudo cross-chroot --leave="Zusatzprogramm mitParameter" pi.img
```

Falls das Zusatzprogramm noch weitere Parameter wie im Beispiel 2 benötigt, muss die komplette Signatur in Anführungszeichen gesetzt werden. Der Pfad zu der virtuellen Root-Partition gefolgt von Rückgabewert von **chroot**, wird von **cross-chroot** in diesem Fall als letzte Perameter angehängt.

Option --version

Gibt die aktuelle Programmversion aus.

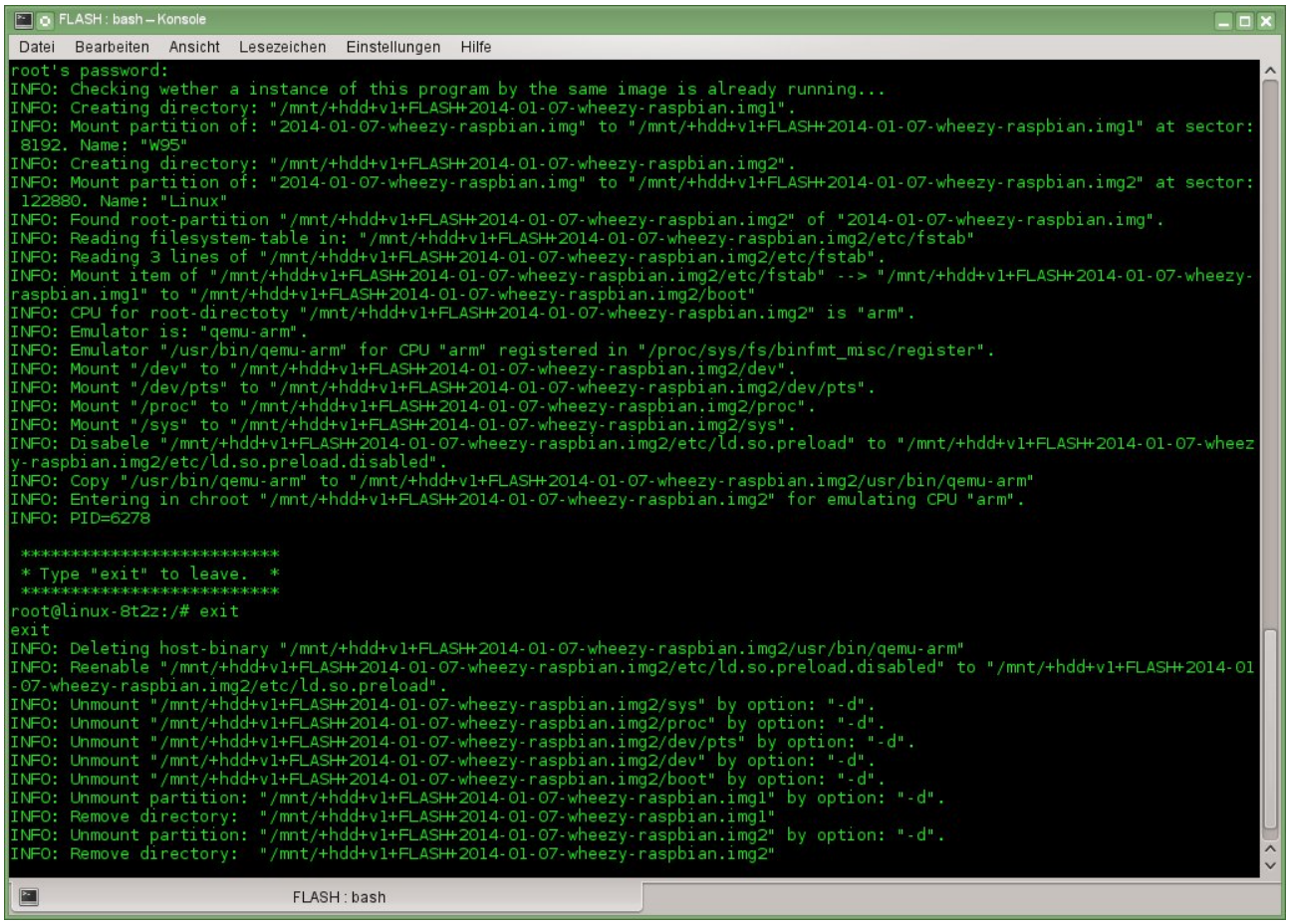
Beispiele für cross-chroot

Beispiel 1

Aufruf mit den Optionen v, m und f und unmittelbarer Beendigung mit der Eingabe von „exit“ in der Chroot-Umgebung:

```
~> sudo cross-chroot -vmf 2014-01-07-wheezy-raspbian.img
```

Resultat:



Beispiel 2:

Aufruf von „uname -m“ in einer normalen (nativen) Linux-Shell:

```
~> uname -m
```

Resultat:

```
x86_64
```

Aufruf von „uname -m“ auf dem selben PC mit einer Chroot-Umgebung für den RaspberryPi:

```
~> sudo cross-chroot 2014-01-07-wheezy-raspbian.img uname -m
```

Resultat:

```
armv7l
```

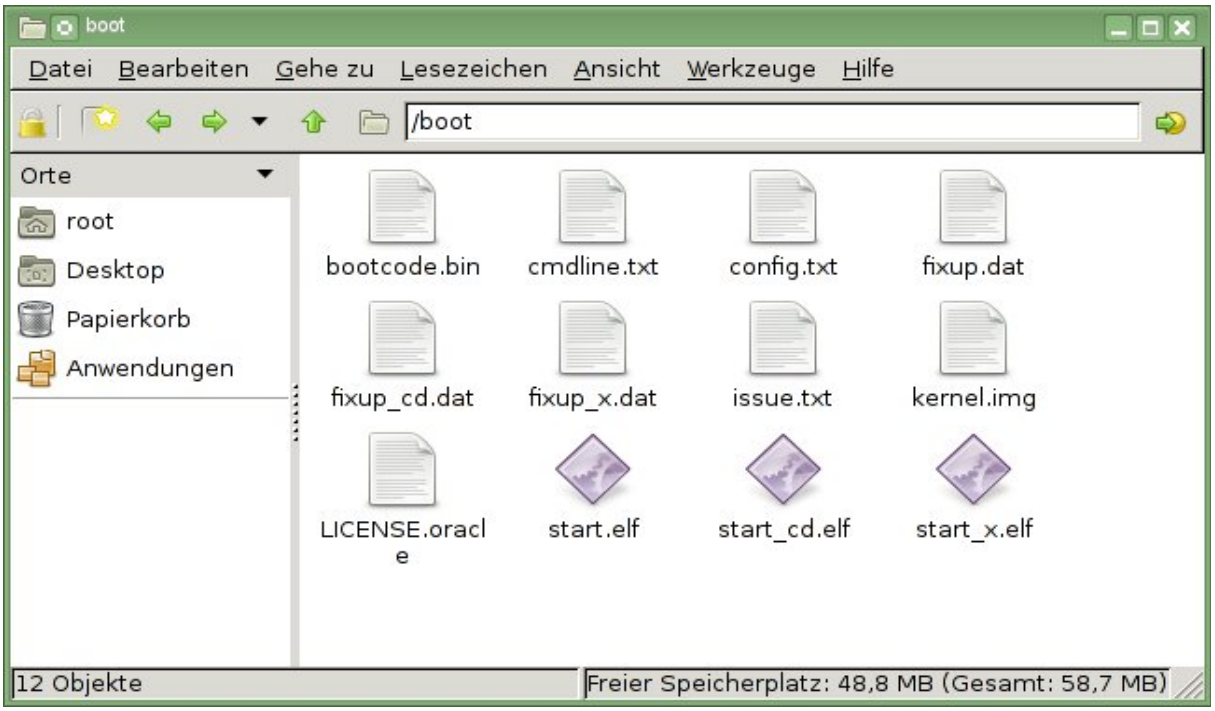
Beispiel 3:

Direkter Aufruf mit der Option -f (Auswertung der fstab) des grafischen LXDE-Dateimanagers „**pcmanfm**“ in der Imagedatei

2014-01-07-wheezy-raspbian.img:

```
~> xsudo cross-chroot -f 2014-01-07-wheezy-raspbian.img pcmanfm
/boot 2>/dev/null
```

Resultat:



iparted (Partitionieringswerkzeug für Image-Dateien)

Dieses Programm ist ein Erweiterungsscript, womit es möglich wird, mittels dem Programm **gparted** (Gnome Partition Editor), Partitionen in einer Imagedatei zu editieren. Ursprünglich ist **gparted** für das Editieren von Partitionen auf Festplatten gedacht. Durch das Erweiterungsscript **iparted,** welches intern **gparted** aufruft, können die Partitionen einer Imagedatei oder eines externen Datenträgers editiert werden, wobei der volle Funktionsumfang von **gparted** genutzt werden kann. Wenn statt einer Imagedatei, die Partitionen auf einem externen Datenträger direkt editiert werden sollen, wird die entsprechende Gerätedatei, welche mit dem externen Datenträger verknüpft ist, zuvor mit den Einträgen in der Datei „**`/etc/dev_permissions.conf`**“ abgeglichen. Wird kein entsprechender Eintrag gefunden, bricht **iparted** die Programmausführung mit einer entsprechenden Fehlermeldung ab.

Sollen die Partitionen einer Imagedatei editiert werden, ändert **iparted** vor und/oder nach dem internen Aufruf von **gparted** die Größe der Imagedatei selber. Bei der Option **-s, --shrink** wird die Größe der Imagedatei an die Obergrenze der höchsten (letzten) Partition angepasst. Um Partitionen in einer Imagedatei zu vergrößern, muss als Aufrufparameter zusätzlich die gewünschte Dateigröße angegeben werden. Statt der Dateigröße kann auch als potentieller externer Ziel-Datenträger, auf das später die Imagedatei (z.B. mit iweite) geschrieben werden soll, eine damit verknüpfte Gerätedatei angegeben werden.

Wenn der zweite Parameter für die Dateigröße fehlt, erscheint eine Dialogbox, wo die Dateigröße eingegeben werden kann. Der voreingestellte Wert ist in diesem Fall die Obergrenze der höchsten Partition, die in der Imagedatei gefunden wurde.

Jedoch sollte man bei der Vergrößerung einer Imagedatei und der darin befindlichen Partitionen, nach folgenden Grundsatz vorgehen:

„So groß wie notwendig und so klein wie möglich.[10]“

Iparted ist in erster Linie dafür gedacht in Linux-Imagedateien genug Platz zu schaffen um Programme und Librarys, z.B. via **cross-chroot** und **apt-get install**, nachzuinstallieren. In der Regel halten sich die Distributionen, welche Imagedateien für embedded Systems zum Download anbieten, an den oben stehenden Grundsatz. Somit sind Partitionen einer frisch heruntergeladenen Imagedatei normalerweise nicht groß genug um z.B. sehr große Bibliotheken wie z.B. **kdelibs5-dev**[11] zu installieren. Für dieses Beispiel sollte eine Imagedatei um etwa 1 GB vergrößert werden, bzw. auf eine Gesamtgröße von mindestens 4 GB.

Was macht iparted?

- Zuerst werden mögliche Optionen ausgewertet.
- Wenn keine Parameter vorhanden sind, erscheint zunächst ein Dateiauswahlmenü in der der eine zu Imagedatei oder Gerätedatei ausgewählt werden kann. Bei der Wahl einer Gerätedatei, die mit einem externen Datenträger verknüpft sein soll, wird

10 Man bedenke, es kann um die 25 Minuten dauern, um eine 8 GB große Imagedatei auf einen externen Datenträger zu kopieren.

11 Wie erfährt die Namen von diversen Paketen? Z.B. Bei Debian-Derivaten mit:

```
sudo cross-chroot -m pi.img apt-cache search kdelib.
```

der gewählte Name mit den Einträgen der Datei **/etc/dev_permissions.conf** abgeglichen, wird kein Eintrag gefunden, wird die Programmausführung mit einer entsprechenden Fehlermeldung beendet. Falls als erster Parameter lediglich ein Verzeichnispfad angegeben wird, Erscheint das Dateiauswahlmenü bei dem dieser Verzeichnispfad bereits eingestellt ist. Wenn als erster Parameter eine Imagedatei oder Gerätedatei angegeben wird, wird das Dateiauswahlmenü übersprungen.

- Wenn der zweite Parameter fehlt, erscheint eine Dialogbox, in der die gewünschte neue Gesamtgröße der Imagedatei angegeben werden kann. Als voreingestellter Standardwert erscheint dort die kleinst mögliche Größe der Imagedatei. Alternativ kann auch eine Gerätedatei angegeben werden, die mit einem externen Datenträger verknüpft sein soll, der später als potentielles Ziel für die Imagedatei dienen kann. Im Fall einer Gerätedatei wird deren Name mit den Einträgen der Datei **/etc/dev_permissions.conf** abgeglichen, wird kein Eintrag gefunden, wird die Programmausführung mit einer entsprechenden Fehlermeldung beendet. Bei einer Gerätedatei wird das maximale Speichervolumen des damit verknüpften Datenträgers ermittelt und damit die Größe der Imagedatei angepasst[12]. Wenn beim Programmaufruf der zweite Parameter ebenfalls vorhanden ist, (gewünschte Dateigröße oder eine Gerätedatei) wird die Eingabedialogbox übersprungen.
- Es wird geprüft, ob die zu editierende Imagedatei oder der externe Datenträger irgendwo im System eingehängt (gemounted) ist. Wenn das der Fall ist, wird die Programmausführung mit einer entsprechenden Fehlermeldung beendet.
- Bei einer zu editierenden Imagedatei wird nun geprüft, ob die gewünschte neue Dateigröße zulässig ist. Ist die Angabe kleiner als die Obergrenze der höchsten in ihr befindlichen Partition, wird das Programm mit einer Fehlermeldung beendet.
- Eine zu editierende Imagedatei wird nun auf die neue Größe vergrößert oder – soweit möglich – verkleinert. Das erfolgt über die Shell-Funktion „**truncate -c** ...“
- Mittels der Shell-Funktion **losetup** werden Loop-Devices als Aufrufparameter für das Programm **gparted** erzeugt, welche mit der Imagedatei oder der Gerätedatei und deren internen Partitionen verknüpft sind.
- Der Gnome Partition Editor **gparted** wird gestartet.
- Nach Beendigung von **gparted**, werden die zuvor generierten Loop-Devices wieder grlöscht.
- Wenn als Aufrufparameter die Option **-s, --shrink** gesetzt wurde, wird nun, im Falle einer Imagedatei, die Dateigröße, auf die Obergrenze der in ihr enthaltenen höchsten Partition minimiert. Zuvor erscheint zu diesem Vorgag eine Dialogbox, in der nachgefragt wird, ob diese Operation wirklich durchgeführt werden soll.

12 Vorsicht, das ist nicht immer sinnvoll. Wir erinnern uns: „So groß wie notwendig und so klein wie möglich.“ Damit Kopierzeit gespart wird.

Aufruf von iparted

Aufruf:

`iparted [options] [Imagefile | Block-device] [Desired entire imagefile-size | possible target block-device]`

Als letzter Parameter kann entweder eine Gerätedatei, welche mit einem externen Datenträger verknüpft ist, dessen Speichergröße für die gewünschte Größe der Imagedatei verwendet wird, oder direkt die gewünschte neue Dateigröße angegeben werden. Fehlt dieser Parameter erscheint ein entsprechender Eingabedialog.

Als Einheiten für die direkte Angabe der neuen Dateigröße werden:

- „B oder b“ für Bytes,
- „K, KiB, KB oder k“ für Kilobyte,
- „M, MiB, MB oder m“ für Megabytes,
- „G, GiB, GB oder g“ für Gigabytes,
- „T, TiB, TB, oder t“ für Terabytes

akzeptiert.

Fehlt die Einheit, wird die Eingabe als Bytes interpretiert. Dezimalbrüche sind zulässig es wird Komma (,) oder Punkt (.) akzeptiert.

Die Einheit muss direkt an die Zahl ohne Leerzeichen angefügt werden,
also z.B. „4,5GiB“ und nicht „4,5 GiB“.

Option -h, --help

Anzeige der Hilfe.

Option -v

Ausgabe von zusätzlichen Informationen auf der Textkonsole.

Option -s, --shrink

Wenn diese Option gesetzt wurde, wird nach der Beendigung des internen Programmaufrufs von **gparted**, bei einer Imagedatei, die Dateigröße auf die Obergrenze der höchsten in ihr enthaltenen Partition reduziert. Zu dieser Maßnahme erscheint vorher eine Yes/No-Dialogbox, welche nochmal nachfragt ob die Aktion durchgeführt werden soll.

Falls die Obergrenze der höchsten Partition bereits der Dateigröße entspricht, erscheint eine kurze Meldung, dass diesbezüglich nichts zu tun ist.

Option -l

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf` in einer Dialogbox angezeigt.

Option -c

Es werden alle erlaubten Gerätedateien aus der Datei `/etc/dev_permissions.conf` in einer Dialogbox angezeigt, die zur Zeit mit einem physikalischen Speichermedium verknüpft sind.

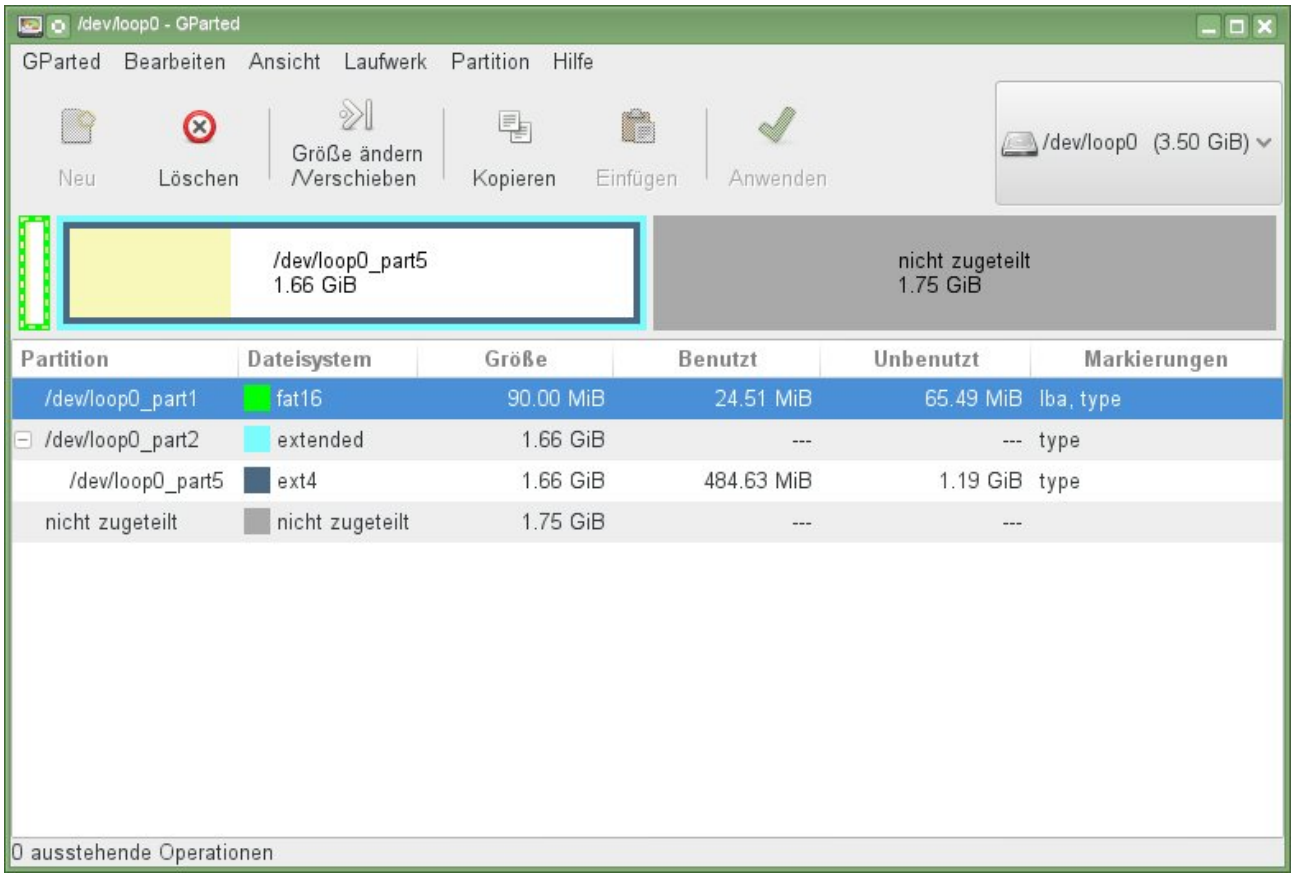
Option --version

Ausgabe der Programmversion.

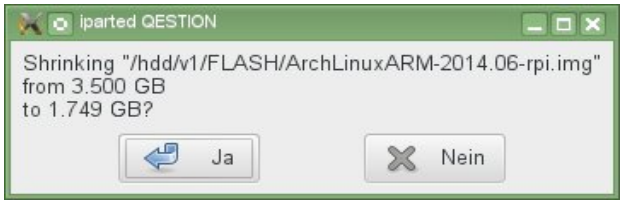
Beispiele

`~> xsudo iparted -s ./ArchLinuxARM-2014.06-rpi.img 3,5GiB`

Führt zu:



Nach Beendigung erscheint – bei Option -s oder --shrink – folgende Nachfrage:



rellink (Relativierer für symbolische Links)

Dieses Programm wandelt in einem Verzeichnis, und – bei gesetzter Option „-r“ oder „--recursive“ – allen Unterverzeichnissen, alle symbolischen Links, welche einen absoluten Verzeichnispfad – also von root „/“ ausgehend – haben, in relative Links – also in „./“ um. Die Voraussetzung ist, dass sich die Orginaldatei, auf die der symbolische Link verweist, im selben Verzeichnis liegt, wie der symbolische Link, andernfalls wird der entsprechende symbolische Link mit einer Warn-Meldung übersprungen.

Wozu braucht man dieses Programm?

In erster Linie ist **rellink** als Hilfsprogramm für die Einrichtung von Cross-Compiler- Tool-Chains gedacht. Es wird (einmalig) benötigt, um z.B. symbolische Links von Bibliotheksdateien, welche in einem virtuellen Root-Verzeichnis liegen, und auf die ein Cross-Compiler/Linker (z.B. arm-linux-gnueabi-gcc) zugreift, zu relativieren.

Das Programm ist auch noch nicht ganz ausgereift, es dient lediglich dazu diverse Sisyphos-Arbeiten zu minimieren, welche bei der Einrichtung von Cross-Compilern anfallen könnten.

Allgemein sind diverse Links auf Bibliotheksdateien, zum Beispiel in **`/usr/lib`**, schon relativiert, dennoch kommt es immer wieder vor, dass dort ein Link doch absolut ist. Das passiert oft, wenn dieser Link über einen Dateimanager angelegt wurde. Im Normalfall, z.B. für einen nativen Compiler/Linker (z.B gcc), ist das vollkommen gleichgültig, da für diesen **`/usr/lib`** auch tatsächlich direkt auf dem Root-Verzeichnis aufsitzt.

Ein Cross-Compiler/Linker greift im allgemeinen auf ein virtuelles Root- Verzeichnis (oder einer z.B. via **imount** eingehängter Partition einer Imagedatei) zu, auf dem sich eine kopierte Linux-Umgebung der Zielplattform befindet. Zum Beispiel könnte der Pfad zu solch einer solchen Bibliotheksdatei so aussehen:

```
/Pfad/zur/virtuellen/Linux-Umgebung/usr/lib/libsigc-1.2.so.5.0.7
```

Der Cross-Compiler/Linker kennt jedoch nur die Datei „`libsigc-1.2.so.5`“ welche als symbolischer Link auf die aktuelle Version dieser Bibliotheksdatei zeigt. In diesem Fall wäre es fatal, wenn dieser Symbolische Link auf:

```
/usr/lib/libsigc-1.2.so.5.0.7
```

zeigen würde, und nicht, wie es richtig wäre, auf:

```
/Pfad/zur/virtuellen/Linux-Umgebung/usr/lib/libsigc-1.2.so.5.0.7
```

Nach der Anwendung des Programms **rellink**, werden die absoluten Pfade solcher „verwaisten“ Links in relative Links umgewandelt.

Das heißt, der absolute symbolische Link, der in diesem Fall auf ein vollkommen falsches Verzeichnis zeigt, wird in einen relativen symbolischen Link umgewandelt. Aus dem absoluten Pfad:

```
/usr/lib/libsigc-1.2.so.5.0.7
```

wird der relative Pfad:

```
./libsigc-1.2.so.5.0.7
```

sofern sich die Orginaldatei im selben Verzeichnis befindet.

Was macht rellink?

- Zuerst werden mögliche Optionen ausgewertet.
- Falls kein weiterer Aufrufparameter angegeben wurde, geht das Programm davon aus, dass der Programmaufruf bereits aus dem Verzeichnis aufgerufen wurde, in dem die symbolischen Links relativiert werden sollen. Andernfalls wechselt das Programm in den als Aufrufparameter angegebenen Pfad.
- Es wird im aktuellen Verzeichnis jeder Eintrag geprüft, ob es sich um einen symbolische Link handelt, und ob dieser ein absoluter ist. Falls das zutrifft, wird überprüft, ob sich die Orginaldatei im selben Verzeichnis befindet. Trifft das auch zu, wird der absolute Pfad des symbolischen Links durch einen relativen Pfad, der auf das aktuelle Verzeichnis zeigt (./), ersetzt.
- Falls die Option `-r` oder `--recursive` (für rekursive Bearbeitung) gesetzt wurde, und im aktuellen Verzeichnis Unterverzeichnisse gefunden werden, wechselt das Programm in diese und ruft sich dort selber auf, um auch dort die gefundenen Links unter den oben stehenden Bedingungen zu relativieren.
- Zum Schluss wechselt das Programm wieder zurück in das Verzeichnis, aus dem es aufgerufen wurde. Falls die Option -s nicht gesetzt wird, wird noch eine kleine Statistik ausgegeben.

Aufruf von rellink

Aufruf:

```
rellink [-option] [base directory]
```

Option -h, --help

Ausgabe der Hilfe

Option -v, --verbose

Ausgabe zusätzlicher Informationen während des Programmablaufs.

Option -s

Wen diese Option angegeben wurde, wird die Ausgabe der Statistik am Programmschluss unterdrückt.

Option -n

Diese Option ist für Testzwecke, es werden keine Links geändert, sondern nur simuliert.

Option -r, --recursive

Bei dieser Option wechselt das Programm in alle Unterverzeichnisse, welche vom angegebenen Basisverzeichnis ausgehen, und ruft sich dort gewissermaßen dort selber wieder auf.

xsudo

Dateiliste

Bekannte Bugs

TODO