Geräte mit Android 3.0 oder 4.0 via MTP in Ubuntu Linux einbinden

15. Dezember 2011 von Christoph Langner | 72 Kommentare

Das Samsung Galaxy Nexus ist ja derzeit in aller Munde. Ich durfte heute auch schonmal ein bisschen mit dem Schmuckstück spielen und muss zugeben, dass mir das Gerät an sich und auch Android 4.0 ziemlich gut gefallen. Eine Sache dämpft jedoch etwas den Enthusiasmus: Aus technischen Gründen kann man den internen Speicher eines Android 3.x oder 4.x Gerätes nicht mehr als USB-Massenspeicher auf einem Computer einbinden. Stattdessen werden die Daten über ein MTP (Media Transfer Protocol) getauftes Protokoll übertragen. Windows-User werden keinen großen Unterschied spüren, da Windows ab Vista das Protokoll von Haus aus im Explorer unterstützt. Für Linux muss man jedoch noch im Terminal zaubern.

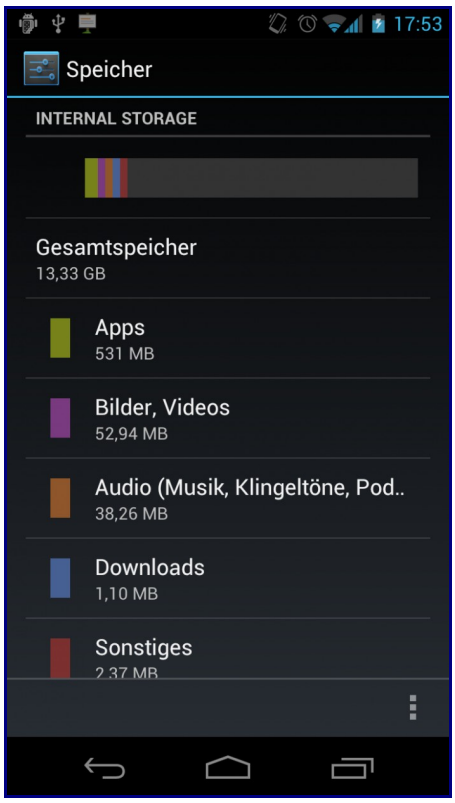
Zu wenig Speicherplatz beim Installieren einer App

Dies liegt an der bisherigen Art und Weise wie Android seinen Speicher einteilte. Android hat den internen Speicher im Prinzip in zwei Partitionen aufgesplittet. Die eine Partition /sdcard dient als Dateiablage für MP3s und sonstige Dateien des Benutzers und ist meist mit FAT32 oder auch FAT16 formatiert. So kann Windows über USB-Massenspeicher die Partition direkt ansprechen.

Kurz mal einen Ausflug in das Thema warum es jetzt keinen USB-Massenspeicher mehr für den internen Speicher gibt. Wer schon länger ein Android-Handy sein eigen nennt, der kennt mit hoher Wahrscheinlichkeit die “Nicht genügend Speicherplatz verfügbar”-Meldung bei der fehlgeschlagenen Installation einer App. Obwohl eigentlich genügend Speicher auf dem Handy verfügbar wäre, klappt die Installation trotz Speichermangel nicht.

YAFFS, Ext4 oder RFS für den internen Speicher

Auf der zweite Partition dagegen liegen mit /system oder /data jedoch Systemverzeichnisse, die ein “besseres” Dateisystem benötigen, dort kommt meist YAFFS, Ext4 oder bei Samsung auch das hauseigene Samsung RFS (Robust File System, eine Erweiterung von FAT mit Journaling) zum Einsatz.



Android besitzt nur noch “einen” Speicher.

Seit Android 3.x steckt jetzt jedoch der komplette interne Speicher auf einer Partition. Das hat den Vorteil, dass nun der gesamte Speicher auch wirklich beliebig genutzt werden kann. Ob man die x GB des Geräts nun mit Bildern, Videos oder Apps zumüllt, ist dem Benutzer überlassen. Die Meldung “Nicht genügend Speicherplatz verfügbar” ist damit Geschichte.

Allerdings heißt eine Partition auch, dass Android den internen Speicher nicht mehr einfach so aushängen kann. Das System hat immer den Finger auf dem Speicher, weil die dort abgelegten Daten auch immer benötigt werden. Somit kann die Freigabe für den exklusiven Zugriff als USB-Massenspeicher nicht mehr erfolgen. Es braucht daher eine neue Methode um Daten zwischen PC und Smartphone/Tablet austauschen.

USB-Massenspeicher nur für SD-Karte

Mit MTP gibt es diese Methode schon länger, doch bislang hat eigentlich kaum ein Hahn danach gedreht. MTP basiert auf dem PTP (Picture Transfer Protocol), das man eventuell schon von billigen Digicams kennt. Diese boten oft nur den Download der Bilder über PTP an, aber eben keinen USB-Massenspeicher. Mit Android 4.0 wird sich diese “Nicht-Nutzung” jetzt allerdings ändern, da die Breite Masse auf einmal mit dieser Funktion in Berührung kommt.

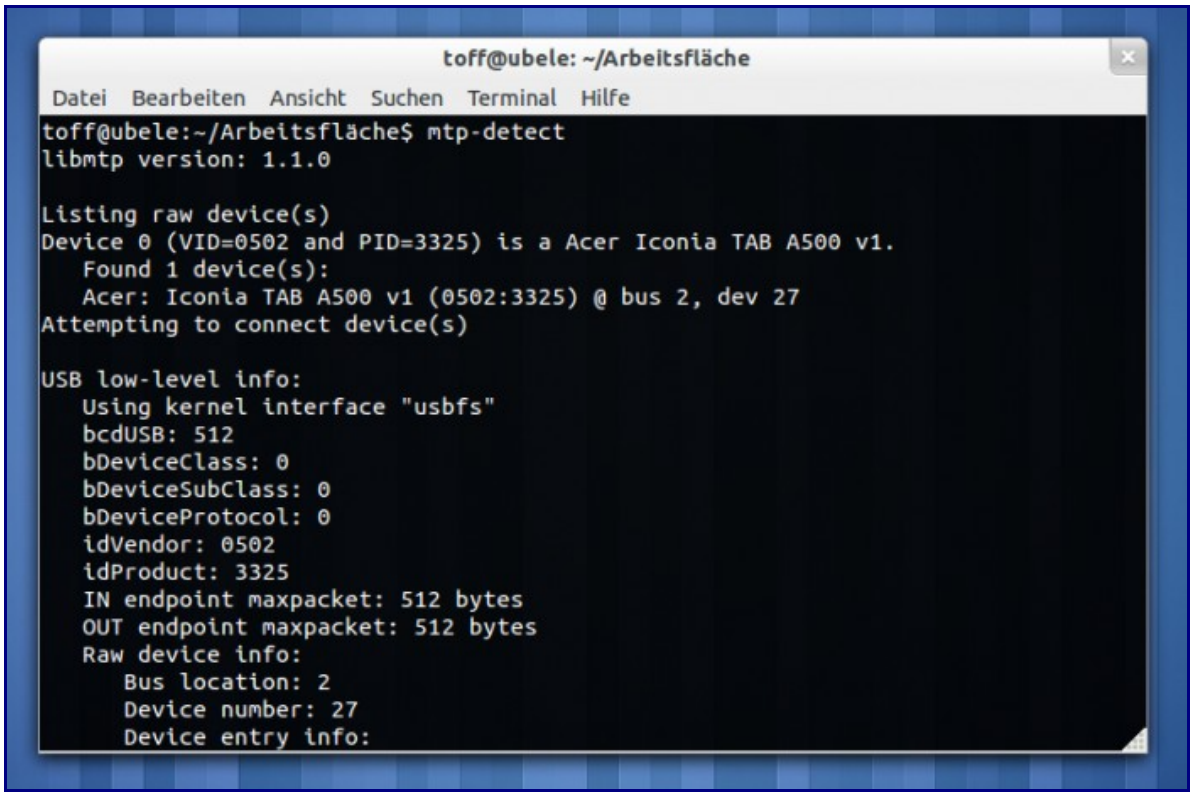
Nach wie vor können Android-Geräte aber auch als USB-Massenspeicher verwendet werden, allerdings *NUR* dann, wenn das Gerät über eine zusätzliche SD-Karte verfügt. Der interne Speicher (das Samsung Galaxy Nexus mit Android 4.0 hat ausschließlich internen Speicher) kann nicht mehr freigegeben werden. Somit kann man dieses Gerät auch nicht mehr als USB-Massenspeicher am Rechner anschließen.

MTP (Media Transfer Protocol) mit Linux

Handarbeit mit der libmtp

Was erwartet nun die Linuxer? Mit libmtp gibt es eigentlich schon eine Bibliothek, die die nötigen Funktionen mitbringt, um MTP auf einem Linux-System nutzen zu können. Es gibt mit MTPFs auch schon eine Methode um MTP über FUSE in das Dateisystem des Linux-Rechners einzubinden. Allerdings ist das Ganz noch nicht ganz anwenderfreundlich, da man ein bisschen im Terminal hantieren muss.

Auf einem aktuellen Ubuntu 11.10 habe ich das Thema zusammen mit einem Acer Iconia Tab (Dort ist gerade Android 3.2 Honeycomb installiert) durchprobiert. Morgen habe ich nochmal ein Samsung Galaxy Nexus mit Android 4.0 zur Verfügung, ich denke aber nicht dass sich diesbezüglich etwas ändern wird.



mtp-detect findet das Acer Iconia Tab.

In Debian oder auch Ubuntu gibt es das Paket mtp-tools, es enthält eine Reihe von Werkzeugen, um Daten via MTP auf ein entsprechendes Gerät zu laden/herunterzuladen/zu löschen usw. Ihr könnt es wie gewohnt über die Paketverwaltung installieren und dann mal ansehen, was es dort alles für Kommandos gibt.

$ sudo apt-get install mtp-tools
$ mtp<tab><tab>
mtp-albumart mtp-files mtp-getplaylist mtp-sendfile
mtp-albums mtp-filetree mtp-hotplug mtp-sendtr
mtp-connect mtp-folders mtp-newfolder mtp-thumb
mtp-delfile mtp-format mtp-newplaylist mtp-tracks
mtp-detect mtpfs mtp-playlists mtp-trexist
mtp-emptyfolders mtp-getfile mtp-reset

Ich will an dieser Stelle nun gar nicht auf alle Befehle eingehen. Interessant sind vielleicht die Befehle *mtp-detect* und *mtp-sendfile*. Über sie könnt Ihr ausprobieren ob Ihr auch tatsächlich über

ein MTP-fähiges Gerät verfügt und auch zum Spaß mal ein Film in die Mediendatenbank des Geräts kopieren.

```
$ mtp-detect
libmtp version: 1.1.0

Listing raw device(s)
Device 0 (VID=0502 and PID=3325) is a Acer Iconia TAB A500 v1.
Found 1 device(s):
Acer: Iconia TAB A500 v1 (0502:3325) @ bus 2, dev 27
Attempting to connect device(s)
USB low-level info:
Using kernel interface "usbfs"
bcdUSB: 512
bDeviceClass: 0
bDeviceSubClass: 0
bDeviceProtocol: 0
[...]
$ mtp-sendfile example.avi Videos
libmtp version: 1.1.0

Device 0 (VID=0502 and PID=3325) is a Acer Iconia TAB A500 v1.

Sending example.avi to Videos
type: avi, 9
Sending file...
Progress: 183585010 of 183585010 (100%)
New file ID: 3407
```

Nach einer kurzen Weile wird der Film kopiert und man kann das USB-Kabel trennen. Es ist nicht nötig das Gerät auszuhängen oder irgendwelche anderen Klimmzüge zu unternehmen. Nun ist es etwas müssig jede einzelne Datei von Hand per mtp-sendfile oder mtp-getfile von Hand hin und her zu friemeln. Von daher gibt es eine Möglichkeit auch MTP-Systeme direkt ins Dateisystem zu mounten. Die Lösung dafür nennt sich MTPfs und FUSE.

Mehr Komfort mit MTPfs

MTPFs findet sich auch schon länger in den Paketquellen von Ubuntu/Debian. Ich hab allerdings nur Ubuntu 11.10 getestet, dort ist eine sehr aktuelle Version enthalten, ich will nicht beurteilen wie das mit älteren Ausgaben des Programms funktioniert.

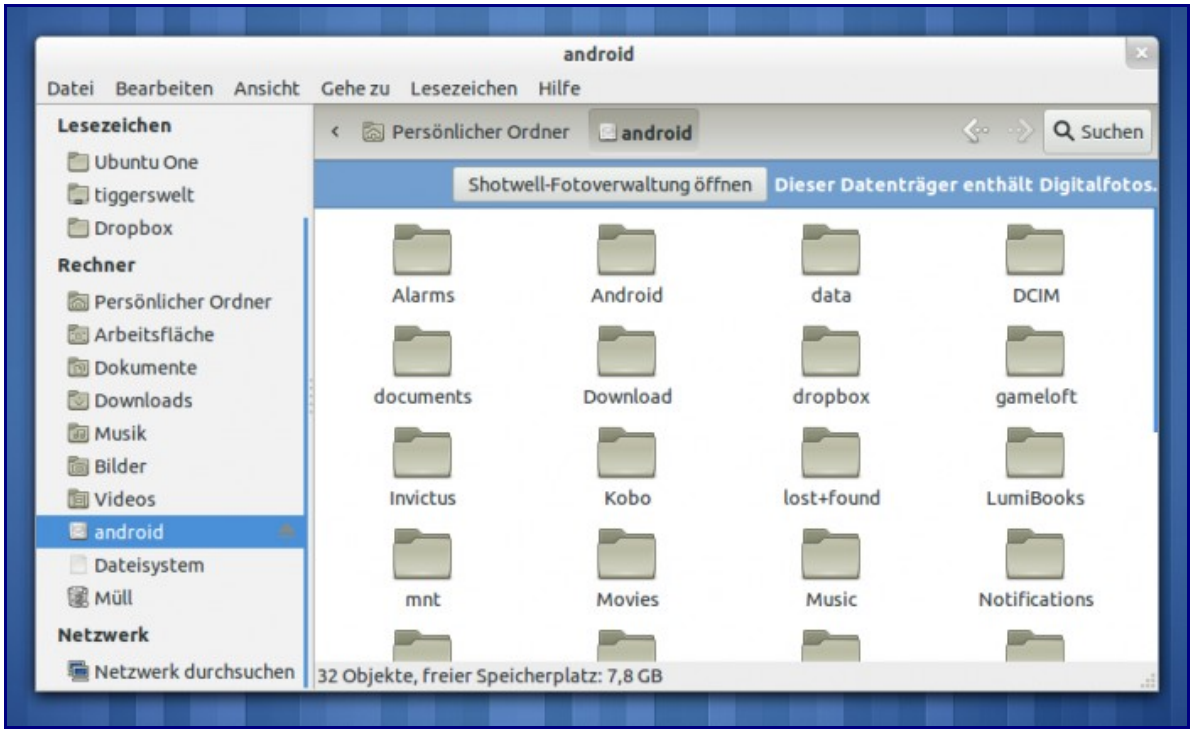
```
$ sudo apt-get install mtpfs
```

Die Anwendung des Ganzen ist dann relativ einfach. Benötigt wird nur ein Mountpunkt (in meinem Beispiel ~/android im Homeverzeichnis des Benutzers und eben das installierte mtpfs-Paket. Im folgenden Beispiel erzeuge ich den Mountpunkt, binde das angeschlossene Android 3.2-Tablet ins Dateisystem ein und hänge es danach – wenn alles Daten übertragen sind – wieder aus.

```
$ mkdir ~/android
$ sudo mtpfs -o allow_other ~/android
$ sudo fusermount -u ~/android # Wieder aushängen nach getaner Arbeit
```

In meinen kurzen Tests ging das alles recht zufriedenstellend. Ich konnte per Nautilus (oder natürlich jeden anderen Dateimanager) transparent alle Daten auf dem Tablet sehen, bearbeiten und auch löschen. Allerdings genehmigte sich das System immer wieder mal Auszeiten, mit ein

bisschen Geduld ließ sich allerdings gut arbeiten. [UPDATE: 15.12: Mit einem Galaxy Nexus und Android 4.0.1 läuft MTPFs super flüssig. Die von mir beobachteten Hänger scheinen am Zusammenspiel zwischen Ubuntu 11.10 und Android 3.x zu liegen.]



Nautilus zeigt die Daten des Android 3.0 bzw. 4.0 Gerätes an.

Auch wenn MTP für Linuxer aktuell ein bisschen Gefrickel, so überwiegen in meinen Augen die Vorteile. Denn wenn ich ehrlich bin, es nervt einfach: Apps nicht mehr aufgrund fehlenden Speichers installieren zu können, obwohl auf dem Gerät eigentlich genügend Speicher frei wäre. Ich bin mir recht sicher, dass die benötigen Bibliotheken fest in Linux-Distributionen verankert werden und man dann die Daten auf Android 3.x/4.x-Geräte auch ohne Frickelei managen kann.