Wiki / dd

Dieser Artikel wurde für die folgenden Ubuntu-Versionen getestet:

Dieser Artikel ist größtenteils für alle Ubuntu-Versionen gültig.

Artikel für fortgeschrittene Anwender

Dieser Artikel erfordert mehr Erfahrung im Umgang mit Linux und ist daher nur für fortgeschrittene Benutzer gedacht.

Zum Verständnis dieses Artikels sind folgende Seiten hilfreich:

1. **Ein Terminal öffnen**
2. **Installation von Programmen**
3. **Bearbeiten von Paketquellen**

Inhaltsverzeichnis

[//media-cdn.ubuntu-de.org/wiki/attachments/10/28/terminal.png] **dd** dient zum bit-genauen Kopieren von Festplatten, Partitionen oder Dateien. "Bit-genaues" Kopieren bedeutet, dass der Datenträger Bit-für-Bit bzw. Byte-für-Byte ausgelesen und beschrieben wird, unabhängig von dessen Inhalt und Belegung. dd funktioniert grundsätzlich mit allen Dateisystemen auf die Ubuntu / Linux zugreifen kann (z.B. ext2/3, reiserfs, vfat, ntfs etc.). Es funktioniert auch mit CD/DVD-Dateisystemen, allerdings nur für Daten-CDs/DVDs.

Dieser Artikel beschreibt die Kommandozeilenversion. Wer lieber mit einer grafischen Benutzeroberfläche arbeitet, kann **gdiskdump** [https://launchpad.net/gdiskdump] nutzen.

Achtung!

- dd wird ohne weitere Rückfragen bzw. Sicherheitsabfragen ausgeführt. Bei unachtsamen Aufrufen könnten evtl. vorhandene Daten überschrieben werden!
- Bevor man eine Partition oder komplette Platte sichert sollte diese **ausgehängt werden**, um sicherzustellen, dass während des Sicherungsvorgangs keine Daten auf die zu sichernde Platte geschrieben werden.
- Zur Übernahme eines bestehenden Systems auf eine **SSD** (Solid-State-Drive) sollte dd nur mit äußerster Vorsicht genutzt werden. In den Standardeinstellungen verwendet dd eine Blockgröße von 512

Bytes, was bei modernen **SSD** zu unnötigen Schreibprozessen führt. Verwendet man unter Benutzung des Parameters bs= eine Blockgrößenangabe die der Blockgröße oder einem Vielfachen davon der **SSD** entspricht, besteht diese Gefahr nicht. Des Weiteren sollte man beachten, dass das **Alignment** eingehalten wird, was ohne weitere Parameter höchstwahrscheinlich nicht der Fall ist.

Installation

Das Programm befindet sich im Paket

- **coreutils**

und ist in jeder Ubuntu-Installation bereits enthalten.

Aufruf

Syntaxaufruf für die Verwendung in der Shell [1]:

```
dd if=Quelle of=Ziel <Optionen>
```

if	Steht für "Input File", kann ein komplettes Gerät (z.B. **/dev/sda**), eine Partition oder eine Datei sein.
of	Steht für "Output File", kann ein komplettes Gerät (z.B. **/dev/sdb**), eine Partition oder eine Datei sein.

dd kann ohne **Root-Rechte** [https://wiki.ubuntuusers.de/sudo/] aufgerufen werden. Man benötigt nur dann Root-Rechte, wenn von einem Gerät bzw. einer Partition gelesen bzw. darauf geschrieben werden soll, auf die nur Root Zugriff hat. Beim Lesen von CD/DVDs muss dd grundsätzlich mit Root-Rechten aufgerufen werden.

Hinweis:

- Wird if bzw. of weggelassen, so liest dd von der Standardeingabe bzw. schreibt auf die Standardausgabe. Dies ist dann nützlich, wenn dd in Kombination mit dem Pipe-Operator genutzt wird.
- dd kann zwar grundsätzlich auch Dateien kopieren, allerdings ist hier in der Regel der Befehl **cp** komfortabler.

Optionen

Der Befehl dd kennt u.a. die folgenden Optionen:

dd - Optionen	
Option	**Beschreibung**
obs=BYTES	Es wird in Blöcken mit der Größe BYTES geschrieben.
ibs=BYTES	Es werden Blöcke der Größe BYTES gelesen.
bs=BYTES	Es werden Blöcke mit der Größe BYTES gelesen und geschrieben. Wird bs als Option benutzt, so ist ibs = obs = bs.

count=BLOCKS	BLOCKS gibt an, wie viele Blöcke mit der durch bs / obs / ids festgelegten Größe gelesen und / oder geschrieben werden.
seek=BLOCKS	BLOCKS gibt an, wie viele Blöcke der mit obs oder bs festgelegten Größe zu Beginn des Schreibvorgangs übersprungen werden.
skip=BLOCKS	BLOCKS gibt an, wie viele Blöcke der mit ibs oder bs festgelegten Größe zu Beginn des Lesevorgangs übersprungen werden sollen.
status=noxfer	Unterdrückt die Ausgabe von Statusinformationen während des Kopiervorgangs. Durch kill -SIGUSR1 <dd-prozess-id> wird der laufende dd manuell dazu veranlasst, Statusinformationen auszugeben.

Für die Angaben BYTES und BLOCKS gilt:

- BYTES muss ganzzahlig sein. Ohne weiteres Suffix wird die Größe der Zahl BYTES in Byte interpretiert.
- BLOCKS muss ganzzahlig sein.

Des Weiteren kennt dd noch verschiedene andere Optionen, insbesondere zum Konvertieren der Daten zwischen Einlesen und Ausgabe. Diese werden bei "normaler" Benutzung eher selten gebraucht, können aber in den **Manpages** [https://wiki.ubuntuusers.de/man/] von dd nachgelesen werden.

Hinweis:

Wenn man keine Blockgröße angibt, verwendet dd eine kleine Standardgröße, was den Datentransfer durch den Overhead sehr langsam macht. Insofern ist es empfehlenswert, z.B. bs=1M anzugeben.

Suffixe für BYTES

Wie oben bereits erwähnt wird die Größe der Zahl BYTES standardmäßig in Byte interpretiert. Diese kann durch Hinzufügen von Suffixes geändert werden.

Suffixe für BYTES	
Suffix	**Multiplikator**
KB	1000 (d.h. 1KB entspricht 1000 Byte)
K	1024 (d.h. 1K entspricht 1024 Byte)
MB	1000000 (= 1000 * 1000, d.h. 1MB entspricht 1000000 Byte)
M	1048576 (= 1024 * 1024, d.h. 1M entspricht 1048576 Byte)
GB	1000000000 (= 1000 * 1000 * 1000, d.h. 1GB entspricht 1000000000 Byte)
G	1073741824 (= 1024 * 1024 * 1024, d,h, 1G entspricht 1073741824 Byte)

Gemäß dem in der Tabelle aufgezeigten Schema gibt es auch die Suffixe TB, T, PB, P, EB, E, ZB, Z, YB, Y - für alle, die wirklich große Datenmengen kopieren müssen.

Die gleichen Suffixe gelten auch für BLOCKS, d.h. z.B. mit **count=1K** werden 1024 Blöcke gelesen/geschrieben, mit **count=1MB** 1000000 Blöcke, usw.

Anwendungen

Achtung!

Da sich die Geräte-Bezeichnungen wie **/dev/sda** nach jedem Bootvorgang ändern können, sind vor der Verwendung von dd stets die aktuellen Gerätedateien zu überprüfen. Dies kann man zum Beispiel mit **blkid** machen.

Einige allgemeine Beispiele

Im Folgenden ein paar allgemeine Beispiele für die Syntax und die Optionen von dd:

- Es wird die komplette fünfte Partition von **/dev/sda** in die erste Partition von **/dev/sdb** kopiert:

```
dd if=/dev/sda5 of=/dev/sdb1
```

- Es werden die ersten zehn 1024 Byte großen Blöcke von der erste Partition von **/dev/sdb** auf die zweite Partition von **/dev/sdc** kopiert:

```
dd if=/dev/sdb1 of=/dev/sdc2 bs=1K count=10
```

- Es werden 2000 Byte große Blöcke von der dritten Partition von **/dev/sda** auf die vierte Partition von **/dev/sda** kopiert, wobei beim Einlesen die ersten 50 Blöcke (in diesen Fall 50 * 2000 = 100.000 Byte) übersprungen werden, d.h. der Lesevorgang fängt bei Byte 100.001 an:

```
dd if=/dev/sda3 of=/dev/sda4 ibs=2KB obs=2KB skip=50
```

Festplatte klonen

Der folgende Befehl klont (kopiert) die komplette Festplatte **/dev/sda** inklusive aller Partitionen, MBR und Partitionstabelle auf die eine zweite Festplatte **/dev/sdb**:

```
dd if=/dev/sda of=/dev/sdb
```

Achtung!

Es sollte darauf geachtet werden, dass die beiden Festplatten gleich groß sind - oder zumindest das Ziel größer. Auch die Sektorengröße (512 Bytes oder Advanced Format mit 4096 Bytes) muss bei Beiden identisch sein.

Falls man plant beide Platten gleichzeitig im selben PC zu betreiben, ist darauf zu achten, dass die **UUIDs** der geklonten Platte geändert werden, da es sonst zu Konflikten kommt.

Eine **miserable Transfergeschwindigkeit** bei großen Festplatten sollte hier mit "bs" zb "bs=1M" vermieden werden, solange die Festplatte keine defekten Blöcke hat. Standardmäßig wird mit 512 Bytes gelesen, bei defekten Blöcken wäre ein "bs=4K" zu empfehlen.

Komprimiert man ein solches Festplattenimage, wie im folgenden Absatz beschrieben, noch zusätzlich mit gzip, so sollte man vorher die Ausgabe von fdisk -l speichern und mit der gesicherten Imagedatei zusammen aufheben. Alternativ kann man die Startpositionen der Partitionen auch – sehr zeitaufwändig – **aus dem gepackten Image auslesen** [https://forum.ubuntuusers.de/topic/partitionen-aus-ge-gziptem-dd-image-auslesen-o/].

Festplatte (sicher) löschen

Achtung!

Alle Daten auf der Festplatte werden unwiderruflich gelöscht!

Der folgende Befehl löscht die komplette Festplatte **/dev/sdX** durch Überschreiben mit Nullen:

```
dd if=/dev/zero of=/dev/sdX
```

Der folgende Befehl löscht die komplette Festplatte **/dev/sdX** durch Überschreiben mit Zufallszahlen (zeitintensiv):

```
dd if=/dev/urandom of=/dev/sdX
```

Experten-Info:

Der Löschprozess kann maßgeblich beschleunigt werden, wenn man den Buffer (internen Zwischenspeicher) der Festplatte ausnutzt. Wie groß dieser für die aktuelle Platte ist, kann mit hdparm -i /dev/sdX herausgefunden werden. Hier ist es der Wert BufferSize=, welcher exakt so (ohne kB) für den dd Parameter bs übernommen werden kann. Zudem kann der Löschprozess in den Hintergrund gelegt werden, um während der Durchführung des Löschvorgangs eine Fortschrittsanzeige auszugeben:

Für die erste Festplatte /dev/sdX mit 8 MiB Puffergröße sieht dass dann so aus:

```
dd if=/dev/zero of=/dev/sdX bs=8M & pid=$!
```

Um nun die Fortschrittsanzeige auszugeben, kann folgendes Kommando auf der selben Konsole eingegeben werden:

```
while true; do kill -USR1 $pid && sleep 1 && clear; done
```

Partitionen klonen

Achtung!

Es sollte darauf geachtet werden, dass die Ziel-Partition gleich groß oder größer als die Quelle-Partition ist.

Der folgende Befehl klont (kopiert) die komplette Partition **/dev/sda1** auf die Partition **/dev/sdb1**:

```
dd if=/dev/sda1 of=/dev/sdb1
```

Man soll sich im Klaren sein, dass dabei alle Partitionsattribute dupliziert werden (Größe, UUID, Label). Alle Geräte mit den dazugehörigen Informationen kann man auflisten:

```
sudo blkid
```

Wenn die Ziel-Partition größer als die Quelle ist, kann man das Zielfilesystem auf die gesamte Partition ausdehnen:

```
sudo resize2fs /dev/sdb1
```

UUID [https://wiki.ubuntuusers.de/UUID/] neu setzen:

```
sudo tune2fs -U random /dev/sdb1
```

Labels [https://wiki.ubuntuusers.de/Labels/] neu vergeben:

```
sudo e2label /dev/sdb1 newlabel
```

Image einer Partition sichern

Der folgende Befehl erstellt ein **Image** [https://de.wikipedia.org/wiki/Speicherabbild] von **/dev/sda1** in die Datei **image_sda1.img**, welche im Heimatverzeichnis gespeichert wird:

```
dd if=/dev/sda1 of=~/image_sda1.img
```

Dies ist die schnellste Methode. Oftmals lässt sich viel Speicherplatz einsparen, wenn das Image zugleich komprimiert wird. Dies bedeutet jedoch größeren Zeitaufwand, was kritisch sein kann, wenn Dateien von einem kränkelnden Datenspeicher gerettet werden sollen.

Der folgende Befehl erstellt ein komprimiertes Image der Partition **/dev/sda1** und speichert dieses in die Datei **image-compress_sda1.img.gz** im Heimatverzeichnis. Durch das Weglassen von of im Befehlsaufruf werden die Daten auf die Standardausgabe geschrieben, wo sie dann per Pipe-Operator an **gzip** [https://wiki.ubuntuusers.de/gzip/] weitergeleitet werden:

```
dd if=/dev/sda1 | gzip > ~/image-compress_sda1.img.gz
```

Hinweis:

Im Regelfall reichen die Standardeinstellungen von gzip aus. Möchte man dennoch die beste Kompressionsstufe, so lautet der zu verwendende Befehl gzip -9. Zu beachten ist, dass bei hohen Kompressionsstufen die Dauer deutlich erhöht - teils sogar vervielfacht - wird, während der Speicherverbrauch nur gering abnimmt [1].

Auch wenn das Image von gzip mit der höchsten Kompressionsstufe komprimiert wird, kann die Ausgabedatei unter Umständen trotzdem sehr groß werden. Man sollte also auf ausreichend Platz auf dem Zieldatenträger achten!

Sofern die zu sichernde Partition nicht verschlüsselt ist, lässt sich die Kompression oft erheblich verbessern, wenn ihr freier Speicherplatz vorab mit **Nullen überschrieben wird** [https://wiki.ubuntuusers.de/Daten_sicher_l%C3%B6schen/#Methode-2]. Bedenken sollte man, dass eine eventuelle

Datenrettung [https://wiki.ubuntuusers.de/Datenrettung/] von versehentlich gelöschten Dateien danach unmöglich wird.

Um das so erzeugte komprimierte Image wieder zurückzusichern, kann man folgenden Befehl verwenden:

```
gzip -cd ~/image-compress_sda1.img.gz | sudo dd of=/dev/sda1
```

Dateigröße des Images begrenzen

Für den Fall, dass zum Beispiel das **Dateisystem** [https://wiki.ubuntuusers.de/Dateisystem/] des Ziellaufwerkes eine **Dateigrößenbeschränkung** [https://wiki.ubuntuusers.de/Dateisystem/#Weitere-Merkmale] hat, besteht die Möglichkeit zum Splitten der Imagedatei. Auf einem FAT32-Laufwerk beispielsweise ist die Dateigröße auf 4 GiB (1 GiB = 1024 * 1024 * 1024 Byte) beschränkt.

In folgendem Beispiel wird das Image an **split** [https://wiki.ubuntuusers.de/split/] übergeben und in Teile von je 3500 MiB (1 MiB = 1024 * 1024 Byte) gespeichert. Hierbei werden die jeweiligen Teile nummerisch beschriftet.

```
dd if=/dev/sda1 | split -d -b 3500M - ~/image_sda1.img.
```

Hinweis:

Nicht den Punkt (**.**) hinter **sda1.img** vergessen! Dahinter steht dann die Folgenummer der Datei. Das Ergebnis sieht dann in diesem Fall so aus:

- image_sda1.img.00
- image_sda1.img.01
- image_sda1.img.02
- image_sda1.img.03

Zurückgespielt wird dann, indem das Image durch **cat** [https://wiki.ubuntuusers.de/cat/] automatisch wieder zusammengefügt und an dd übergeben wird.

```
cat ~/image_sda1.img.* | dd of=/dev/sda1
```

MBR: Boot-Loader und Partitionstabelle sichern

Der MasterBootRecord (MBR) beherbergt den Boot-Loader, die Partitionstabelle und die MBR-Signatur. Der MBR ist exakt 512 Bytes lang und liegt am Beginn der Festplatte. Der Boot-Loader belegt die ersten 446 Bytes des MBR, dann folgen die Partitionstabelle (64 Bytes) und die MBR-Signatur, und, Achtung, zumindest **GRUB** [https://wiki.ubuntuusers.de/GRUB/](**2** [https://wiki.ubuntuusers.de/GRUB_2/]) nutzt je nach Konfiguration meist noch einige weitere Sektoren im sog. verborgenen Bereich vor der ersten Partition.

Zur Sicherung ist ein geeignetes Medium notwendig. Nutzer einer LiveCD (z. B. Ubuntu Installations-CD) müssen zunächst ein Medium verfügbar machen:

```
sudo fdisk -l
```

zeigt die Bezeichnungen der eigenen Festplatten an und dient als Orientierunghilfe für folgende Kommandos.

Nun erstellt man einen Ordner im Dateisystem der Live-CD und hängt dort eine Partition ein, auf welcher die Sicherung des MBR erstellt wird.

```
sudo mkdir ~/sda3
sudo mount /dev/sda3 ~/sda3
cd ~/sda3
```

Es kann sich hierbei auch um einen USB-Stick, eine Netzwerkfreigabe oder ein anderes Medium handeln, auf welches man jederzeit Zugriff hat. Jetzt kann mit dem eigentlichen Sichern begonnen werden.

Achtung!

Die Sicherung der Partitionstabelle mittels dd ist mit Vorsicht anzuwenden, da:

1. mit dem MBR nur die primären Einträge der Partitionstabelle (Bytes 446..509) gesichert werden. Die Einträge zu den logischen Partitionen stehen in den kaskadierten BRs (BootRecords) der erweiterten Partition, und fehlen damit hier komplett.
2. beim Einsatz einer **GPT** (**GPT**) überhaupt keine Sicherung der Partitionstabelle erfolgt. Außerdem wird der **Bootloader in eine eigene Boot-Partitionen** installiert.

Um immer auf der sicheren Seite zu sein, empfiehlt es sich nach jeder Partitionsänderung die entsprechenden Tabellen neu zu sichern. Bei der MBR-Partitionstabelle kann man dazu das Programm **sfdisk** und bei der GUID-Partitionstabelle das Programm **sgdisk** nutzen. Wer es noch einfacher möchte, der kann das Skript **Partitionstabellen sichern** dazu nutzen. Spielt man eine alte MBR-Sicherung (mit alter und somit falscher Partitionstabelle) zurück, kann man auf die komplette Platte höchst wahrscheinlich nicht mehr zugreifen.

Hinweis:

Die folgenden Anweisungen zum Sichern des MBR sollten nur als Muster der Möglichkeiten von dd verstanden werden.

Mit dem folgenden Befehlsaufruf würde der Boot-Loader der Festplatte **/dev/sda** als Datei **bootloader_sicherung** im aktuellen Verzeichnis gesichert. Die Partitionstabelle (Bytes 446..509) und die MBR-Signatur sind in dieser Sicherung nicht enthalten:

```
sudo dd if=/dev/sda of=bootloader_sicherung bs=446 count=1
```

Der folgende Befehl sichert den gesamten MBR (inklusive Partitionstabelle) der Festplatte **/dev/sda** als Datei **mbr_sicherung** im aktuellen Verzeichnis:

```
sudo dd if=/dev/sda of=mbr_sicherung bs=512 count=1
```

Bei installiertem **Boot-Manager** [https://wiki.ubuntuusers.de/Dualboot/], z.B. **GRUB 2** [https://wiki.ubuntuusers.de/GRUB_2/], sollte man ggf. auch den sog. verborgenen Bereich hinter dem MBR, falls dieser (Regelfall) **dafür verwendet wird** [https://wiki.ubuntuusers.de/GRUB-Umgebung_analysieren/], mitsichern und vorher schauen, wie viele Sektoren vor der ersten Partition frei sind:

```
sudo fdisk -lu
```

Die erste Zahl in der Spalte Anfang zeigt den Startsektor der ersten Partition. Bei heutigen Festplatten beginnt die erste Partition meist bei Sektor 63. Dann schaut man noch wie viele Bytes ein Sektor hat – meist 512 – und passt den Befehl entsprechend an:

```
sudo dd if=/dev/sda of=mbr+grub_sicherung bs=512 count=63
```

Die Partitionen von **SSDs** [https://wiki.ubuntuusers.de/SSD/] sollten an ganzen Megabytes **ausgerichtet** [https://wiki.ubuntuusers.de/SSD/Alignment/] sein. Also sollte die erste Partition einer SSD bei 1 MiB (meist 2048 Sektoren · 512 Bytes) beginnen und der Befehl lautet

```
sudo dd if=/dev/sda of=mbr+grub_sicherung bs=1M count=1
```

Eine Sicherung des Boot-Loaders wird mit

```
dd if=bootloader_sicherung of=/dev/sda bs=446 count=1
```

zurückgespielt. Dieses Kommando kann auch unter Verwendung einer kompletten Sicherung des MBR bzw. obiger **mbr+grub_sicherung** verwendet werden: Es wird nur der Boot-Loader (Bytes 0..445) wiederhergestellt, die momentane Partitionstabelle (Bytes 446..509) und die MBR-Signatur bleiben dann in jedem Fall erhalten.

Will man den kompletten MBR (also inklusive Partitionstabelle) zurücksichern, so lautet der Befehl wie folgt:

```
dd if=mbr_sicherung of=/dev/sda bs=512 count=1
```

Will man im Fall eines installierten **Boot-Managers** [https://wiki.ubuntuusers.de/Dualboot/] zusätzlich auch diesen (falls im sog. verborgenen Bereich hinter dem MBR abgelegt) zurücksichern (unter Erhalt des zuvor rückgesicherten Boot-Loaders und der Partitionstabelle), so lautet der Befehl wie folgt (falls die erste Partition bei Sektor 63 beginnt, siehe detaillierteren Hinweis weiter oben):

```
dd if=mbr+grub_sicherung of=/dev/sda bs=512 skip=1 seek=1 count=62
```

Mit dd erstellte Images einbinden

Image einer Partition einbinden

Ein mit dd erstelltes Image lässt sich als Loop-Device mit dem Befehl **mount** [https://wiki.ubuntuusers.de/mount/] einbinden. So kann auf das Image wie auf ein normales Laufwerk zugegriffen werden. Dazu erstellt man als erstes ein Image, hier z.B. vom Device **/dev/sda1**, gespeichert in der Datei **loop_image.img** im Heimatverzeichnis:

```
dd if=/dev/sda1 of=~/loop_image.img
```

Dann erzeugt man einen Einhängpunkt, z.B. **/media/loop_mount**:

```
sudo mkdir /media/loop_mount
```

Jetzt kann man das mit dd erzeugte Image mit **mount** einbinden:

```
sudo mount -o loop ~/loop_image.img /media/loop_mount
```

Nun kann man auf alle Dateien, Verzeichnisse etc. des Images wie auf ein reguläres Laufwerk zugreifen. Nach der Benutzung muss man das Image dann wieder mit **umount** aushängen:

```
sudo umount /media/loop_mount
```

Bei Bedarf kann das (bearbeitete) Image jetzt auch wieder zurück gesichert werden.

Partition aus einem Image der gesamten Platte einbinden

Hat man nicht nur eine Partition, sondern die gesamte Festplatte inclusive MBR gesichert, braucht man den Offset der jeweiligen Partition. Diesen kann man mit dem Befehl

```
sudo fdisk -l /Pfad/zum/Image.img
```

herausfinden. Die Ausgabe sieht bei 3 primären Partitionen ungefähr so aus:

```
Platte /Pfad/zum/Image.img: 0 MByte, 0 Byte
255 Köpfe, 63 Sektoren/Spuren, 0 Zylinder, zusammen 0 Sektoren
Einheiten = Sektoren von 1 × 512 = 512 Bytes
Disk identifier: 0xd53d826f

    Gerät            boot. Anfang      Ende         Blöcke     Id  System
/Pfad/zum/Image.img1   *          63   104872319    52436128+   7  HPFS/NTFS
Partition 1 hat unterschiedliche phys./log. Enden:
     phys=(1023, 254, 63) logisch=(6527, 254, 63)
/Pfad/zum/Image.img2       104872320   109065284    2096482+   82  Linux Swap /
Solaris
Partition 2 hat unterschiedliche phys./log. Anfänge (nicht-Linux?):
     phys=(1023, 0, 1) logisch=(6528, 0, 1)
Partition 2 hat unterschiedliche phys./log. Enden:
     phys=(1023, 254, 63) logisch=(6788, 254, 63)
/Pfad/zum/Image.img3       109065285   156296384    23615550   83  Linux
Partition 3 hat unterschiedliche phys./log. Anfänge (nicht-Linux?):
     phys=(1023, 0, 1) logisch=(6789, 0, 1)
Partition 3 hat unterschiedliche phys./log. Enden:
     phys=(1023, 254, 63) logisch=(9728, 254, 63)
```

Der Wert hinter der entsprechende Partition unter Anfang, ist der Offset, dieser muss jedoch noch mit der weiter oben angegebenen Sektorgröße multipliziert werden (hier 512). Der Offset für die 3. Partition wäre also 109065285 * 512 = 55841425920. Nun Folgt der Mountbefehl mit dem entsprechenden Offset (hier wieder am Beispiel der 3. Partition):

```
sudo mkdir /media/loop_mount # Verzeichniss anlegen
sudo mount -o loop,offset=55841425920 /Pfad/zum/Image.img /media/loop_mount
```

Zum Schluss wird das Image wieder freigegeben mit:

```
sudo umount /media/loop_mount
```

Des Weiteren gibt es ein **fertiges Skript** [https://wiki.ubuntuusers.de/mount/#Festplatten-Image], mit dem mit dd erstellte Image-Dateien komfortable eingebunden werden können.

Image im Netzwerk speichern

Ein mit dd erstelltes Image muss nicht zwangsläufig lokal gespeichert werden, sondern kann auch auf einen anderen Rechner im Netzwerk gesichert werden. Im folgenden Beispiel wird mit dd ein Image von **/dev/sda1** erstellt, welches dann **ssh-verschlüsselt** [https://wiki.ubuntuusers.de/SSH/] auf den Rechner mit der IP-Adresse 192.168.0.100 übertragen und dort im Verzeichnis **/home/BENUTZER** in der Datei **image_sda1.img** gespeichert wird. Damit dies funktioniert, muss "BENUTZER" ein Benutzerkonto auf dem entsprechenden Rechner haben und man selbst die notwendigen **Rechte** [https://wiki.ubuntuusers.de/Rechte/], um dort zu schreiben. Der Befehlsaufruf lautet:

gzip-komprimiert und ssh-verschlüsselt

```
dd if=/dev/sda1 | gzip -9 - | ssh user@192.168.0.100 "cat >
/home/BENUTZER/image_sda1.img.gz"
```

Um das Image zurückzusichern (z.B. auf den Rechner mit der IP-Adresse 192.168.0.50), gibt man folgenden Befehl ein:

```
ssh user@192.168.0.50 "cat /home/user/image_sda1.img.gz" | gunzip -c - | dd
of=/dev/sda1
```

bzip2-komprimiert und nicht verschlüsselt

Alternativ mit bzip komprimiert, aber im Transfer nicht:

- Auf dem Zielrechner:

  ```
  netcat -l -p 5555 | dd of=/home/user/image_sda1.img bs=16065b
  ```

- Auf dem Quellrechner:

  ```
  dd if=/dev/sda1 bs=16065b | pv | bzip2 -1 | netcat ZielIP 5555
  ```

  Zur Verwendung des Befehls **pv** siehe **Fortschrittsanzeige**.

FTP mit gzip Komprimierung und nicht verschlüsselt

Erstellen eines Images über FTP:

```
dd if=/dev/sda bs=4k | gzip -9 - | ncftpput -c -V -u FTPUSER -p FTPPASSWORT
FTPSERVER /FTPPATH/NAME.img.gz
```

Und zum Einspielen vom erstellten Images:

```
ncftpget -c -V -u FTPUSER -p FTPPASSWORT FTPSERVER /FTPPATH/NAME.img.gz | gunzip -
c - | dd of=/dev/sda bs=4k
```

Fortschrittsanzeige

Sehr häufig wird eine Fortschrittsanzeige gewünscht, um den Status und die verbleibende Restdauer einer Aktion mit dd nachvollziehen zu können.

Seit Version 8.24

In Ubuntu 16.04 ist dd in der Version 8.25 enthalten. Seit Version 8.24 ist es mit dem Parameter **status** möglich den Fortschritt anzuzeigen. Der Befehl dazu könnte so aussehen:

```
dd if=/dev/XXX of=/dev/XXX status=progress
```

Einmalige oder regelmäßige Abfrage mittels Senden des Signals -USR1

Informationen zum ermitteln von Prozessen siehe **ps** [https://wiki.ubuntuusers.de/Shell/ps/], **pgrep** [https://wiki.ubuntuusers.de/pgrep/] oder **pidof** [https://wiki.ubuntuusers.de/Shell/pidof/] und zum senden von Signalen **kill** [https://wiki.ubuntuusers.de/Shell/kill/], **pkill** [https://wiki.ubuntuusers.de/pkill/] oder **killall** [https://wiki.ubuntuusers.de/Shell/killall/].

Wenn das dd-Kommando einmal abgesetzt wurde, wünscht man sich bei größeren Kopiervorgängen eine Kontrollmöglichkeit über den Fortschritt. Dies erreicht man indem man dem dd-Prozess das Signal -USR1 sendet.

ps und kill

In einem zweiten Terminal, ermittelt man die Prozessnummer, z.B. mit

```
ps -a
```

und setzt ein Signal -USR1 ab. (bei Lucid Lynx mit *sudo*).

```
kill -USR1 <prozessnummer>
```

Die bisher kopierte Datenmenge wird dann in dem Terminal angezeigt, in dem dd gestartet wurde. Mit einer Kombination aus dd und einer Schleife kann man dies auch automatisieren.

```
dd if=/dev/XXX of=/dev/XXX & ddpid=$! ; while [ $(ps -ao pid | grep $ddpid) ]; do
kill -SIGUSR1 $ddpid; sleep 10; done
```

pkill

Sofern nur ein Prozess mit dem Namen dd läuft kann man dies auch über den Prozessnamen.

```
pkill -USR1 -x dd
```

In einer Schleife kann das so aussehen:

```
dd if=/dev/XXX of=/dev/XXX & while [ ! $(pkill -USR1 -x dd) ]; do sleep 10 ; : ;
done
```

oder alternativ:

```
watch -n10 pkill -USR1 -x dd
```

pv

Als Alternative zur oben beschriebenen Vorgehensweise kann man auch den Befehl **pv** [https://wiki.ubuntuusers.de/pv/] verwenden, um sich den Fortschritt anzeigen zu lassen. Dabei wird pv mittels einer Pipe dazwischengeschaltet. Um sich anzeigen zu lassen, wie weit der Vorgang fortgeschritten ist und wann er voraussichtlich beendet sein wird, muss man allerdings die Größe der Partition bzw. der Festplatte kennen.

Live USB-Stick erstellen

Mit dd lässt sich auch auf einfachste Art und Weise ein Live USB-Stick (als Ersatz für eine Live-CD) erstellen. Zwingende Voraussetzung ist allerdings ein entsprechendes Hybrid-ISO-Image. Die Live-ISO-Images von Ubuntu und seinen Varianten sind erst ab **Ubuntu 11.10** [https://wiki.ubuntuusers.de/Oneiric/] Hybrid-ISO-Images. Auch viele andere Linux-Distributionen stellen diese zur Verfügung.

Im folgenden Beispiel wird davon ausgegangen, dass der USB-Stick als /dev/sdx erkannt wurde und nicht **eingebunden** [https://wiki.ubuntuusers.de/mount/#Dateisysteme-aushaengen] ist (aber bitte nicht auswerfen bzw. *"sicher entfernen"*):

```
sudo dd if=hybrid_iso_image.iso of=/dev/sdx bs=1M && sync
```

Dabei beachten, dass der Pfad zum USB-Stick angegeben wird (z.B. sdx) und nicht der zu einer Partition (z.B. sdx1). Den Parameter bs=1M kann man auch weglassen, aber er beschleunigt den Kopiervorgang. Weitere Tipps zu bootbaren USB-Medien findet man im Artikel **Live-USB** [https://wiki.ubuntuusers.de/Live-USB/].

Links

- **gzip compression rates** [http://compressionratings.com/i_gzip.html]
- **Gdiskdump – powerful hard disk cloning tool** [http://gamblisfx.com/gdiskdump-powerful-hard-disk-cloning-tool/] - GUI für dd. Blogbeitrag, 04/2014
- **Sicheres Löschen: Einmal überschreiben genügt** [https://www.heise.de/-198816] - heise News, 01/2009
- **dd mit Statusanzeige** [http://o-o-s.de/dd-mit-statusanzeige/79] - Blogbeitrag, 01/2007

Diese Revision [https://wiki.ubuntuusers.de/dd/a/revision/937363/] wurde am 9. März 2017 11:10 von **4atn1** erstellt.
Die folgenden Schlagworte wurden dem Artikel zugewiesen: **Klonen** [https://wiki.ubuntuusers.de/wiki/tags/Klonen/], **Image** [https://wiki.ubuntuusers.de/wiki/tags/Image/], **Datensicherung** [https://wiki.ubuntuusers.de/wiki/tags/Datensicherung/], **Shell** [https://wiki.ubuntuusers.de/wiki/tags/Shell/], **Sicherheit** [https://wiki.ubuntuusers.de/wiki/tags/Sicherheit/]