

SelfLinux-0.13.1



## Das Partitionieren



Autor: Arnulf Pelzer ([webmaster@arnulfpelzer.de](mailto:webmaster@arnulfpelzer.de))  
Formatierung: Matthias Hagedorn ([matthias.hagedorn@selflinux.org](mailto:matthias.hagedorn@selflinux.org))  
Lizenz: GPL

---

# Inhaltsverzeichnis

## 1 Das Partitionieren

## 2 Programme zum Partitionieren

- 2.1 Partitionieren mit cfdisk
- 2.2 Partitionieren mit qtparted

## 3 Formatieren der Festplatte

- 3.1 Erstellen eines ext2 Dateisystems
- 3.2 Erstellen eines ext3 Dateisystems
- 3.3 Erstellen eines ReiserFS Dateisystems
- 3.4 Erstellen eines Reiser4 Dateisystems
- 3.5 Erstellen eines XFS Dateisystems
- 3.6 Erstellen eines JFS Dateisystems
- 3.7 Erstellen eines swap Bereichs

## 1 Das Partitionieren

Um Ordner und Dateien auf Festplatten abzuspeichern oder zu lesen ist es notwendig, die Festplatte in einen Zustand zu versetzen, der dies ermöglicht. Der erste Schritt ist das Partitionieren, der zweite das Formatieren. Dieser Text soll Ihnen helfen mit dieser Hürde umzugehen. Das Partitionieren ist das Aufteilen einer Festplatte in mehrere Teile, sogenannte Partitionen. Diese sind voneinander unabhängig und können verschiedene Dateisysteme enthalten. Partitionen entsprechen Schubladen in einem Schrank. Zieht man eine heraus und nimmt etwas heraus, leert sie oder ändert den Inhalt, werden die anderen Schubladen davon nicht betroffen. Bei PCs gibt das PC-BIOS (**Basic Input Output System**) vor, wieviele Partitionen verwendet werden dürfen. Für PC-Festplatten sind die eigentlichen Partitionen **Primäre Partitionen**, davon können Sie aber nur vier anlegen. Um diese Einschränkung zu umgehen, wurden erweiterte (extended) Partitionen eingeführt. Eine primäre Partition kann als erweiterte Partition in mehrere logische Partitionen unterteilt werden. Vom  [BIOS](#) gibt es keine Beschränkungen für die Anzahl logischer Partitionen. Es darf aber nur eine erweiterte Partition pro Festplatte existieren. Linux limitiert die Anzahl der Partitionen pro IDE-Festplatte auf 63 Partitionen, drei benutzbar als primäre und 60 als logische Partitionen. Bei SCSI-Festplatten 15, drei benutzbar als primäre und zwölf als logische Partitionen.

Bei älteren PCs oder BIOS-Versionen sei darauf hingewiesen, dass die **Boot**-Partition, jene die den Kernel enthält, innerhalb der ersten 1024 Zylinder liegen muss. Da, in der Regel, die **Boot** und die **Root** Partition identisch sind, müssen Sie darauf achten dass die **Root**-Partition innerhalb der ersten 1024 Zylinder der Festplatte untergebracht ist.

Bei neueren BIOS-Versionen die **Enhanced Disk Drive Support** verwenden gibt es diese Einschränkungen nicht mehr. [lilo](#), [grub](#) und andere  [Bootmanager](#) greifen über das BIOS auf int 0x13 (Unterstützung für grosse Festplatten) um den [Kernel](#) in den Speicher zu laden. Sobald Linux gestartet ist, wird nicht mehr auf das BIOS zurückgegriffen, da Linux eigene Routinen für den Festplattenzugriff verwendet. GNU/Linux Systeme sollten den [Filesystem Hierarchy Standard](#) (FHS) unterstützen, dies ermöglicht es dem Nutzer Dateien an bestimmten Orten im Dateisystem zu finden. Das Root Verzeichnis wird durch einen einfachen Schrägstrich - slash (/) - dargestellt.

|       |                                                   |
|-------|---------------------------------------------------|
| /     | Wurzel-Verzeichnis                                |
| /bin  | wesentliche Programme                             |
| /boot | statische Dateien des Bootmanagers                |
| /dev  | Schnittstellen zu Geräten                         |
| /etc  | Rechnerabhängige Konfigurationsdateien            |
| /home | Heimatverzeichnisse der Benutzer                  |
| /lib  | Wesentliche Systembibliotheken                    |
| /mnt  | Einhänge Punkte für temporäre Dateisysteme        |
| /proc | Virtuelles Verzeichnis für Systeminformationen    |
| /sys  | Virtuelles Verzeichnis für Systeminformationen    |
| /root | Heimatverzeichnis des Systemadministrators (root) |
| /sbin | wesentliche Systemprogramme                       |
| /tmp  | temporäre Dateien                                 |
| /usr  | sekundäre Dateisystemhierarchie                   |
| /var  | Veränderliche Dateien                             |
| /opt  | zusätzliche Software Pakete                       |

Eine weitere Partition, die in vielen GNU/Linux Systemen zu finden ist, nennt man **swap**. Diese Partition dient dazu Speicherbereiche des Hauptspeichers auszulagern, um weitere Programme auszuführen oder ausgeführten Programmen mehr als den im System vorhandenen Speicher zugänglich zu machen. Diese Funktion ist bei

Rechnern mit wenig Hauptspeicher sehr von Vorteil, da es ermöglicht  [OpenOffice](#) oder  [KDE](#) zu starten ohne eigentlich den notwendigen Speicher dafür zur Verfügung zu haben. Es ist aber nur eingeschränkt sinnvoll, da durch das Auslagern sehr viel Zeit vergeht und ein zügiges Arbeiten nicht möglich ist.

Auf meinen privaten Rechnern mit 512 MB Ram und jeweils 1024 MB **swap**-Partition wird sehr selten der **swap**-Bereich der Festplatte angefordert.

Bei Rechnern mit anderen Aufgaben kann dies aber ganz anders aussehen. Sie sollten aber immer eine **swap** Partition anlegen, da dies nur von Vorteil sein kann.

Ein GNU/Linux System braucht nur eine Partition um lauffähig zu sein.

Auf meinem Laptop sieht die Partitionierung so aus:

```
Platte /dev/hda: 30.0 GByte, 30005821440 Byte
255 Köpfe, 63 Sektoren/Spuren, 3648 Zylinder
Einheiten = Zylinder von 16065 × 512 = 8225280 Bytes
```

| Gerät     | boot. | Anfang | Ende | Blöcke    | Id | System     |
|-----------|-------|--------|------|-----------|----|------------|
| /dev/hda1 |       | 1      | 1721 | 13823901  | 83 | Linux      |
| /dev/hda2 |       | 1722   | 1848 | 1020127+  | 82 | Linux swap |
| /dev/hda3 |       | 1849   | 3647 | 14450467+ | 83 | Linux      |

```
/dev/hda1 ist die boot und root Partition
/dev/hda2 ist die swap Partition
/dev/hda3 ist die home Partition
```

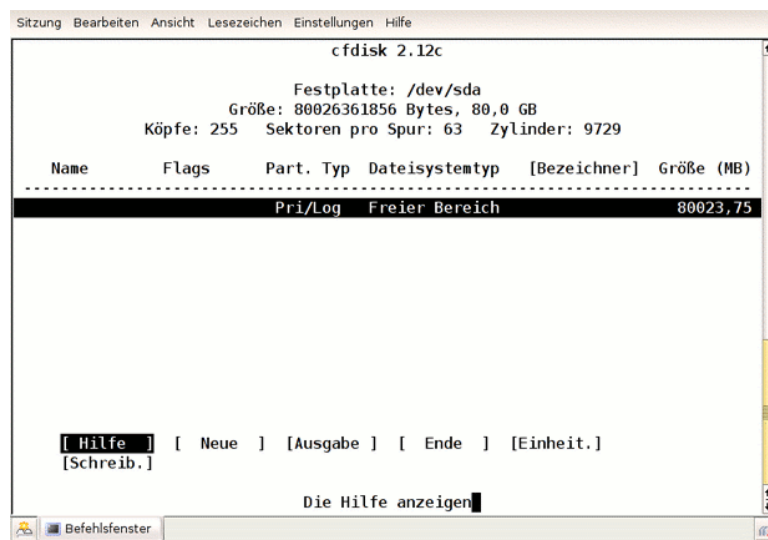
Da es zu jeder Aufgabe für einen Computer verschiedene Partitionierungs-Philosophien gibt, masse ich mir hier keine Empfehlung an.

## 2 Programme zum Partitionieren

Es gibt unter GNU/Linux verschiedene Werkzeuge, um Festplatten zu partitionieren. **fdisk**, **parted** und die hier vorgestellten Programme **cdisk** und **qtparted**.

### 2.1 Partitionieren mit cfdisk

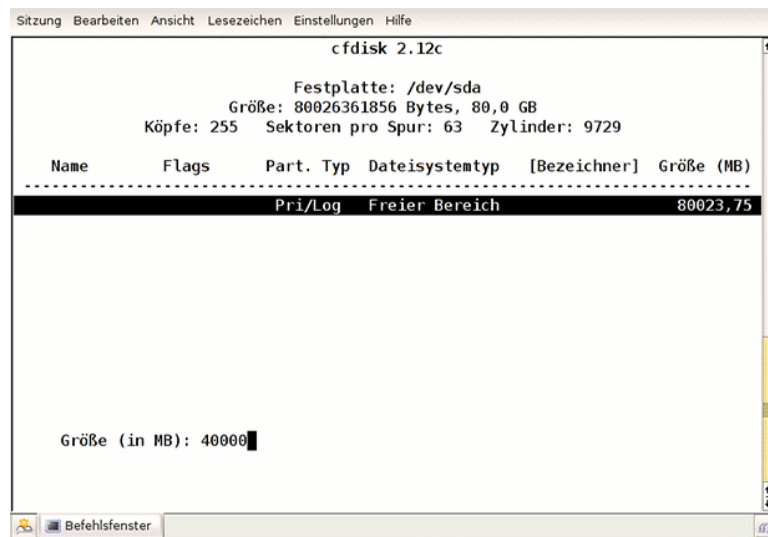
Auf diesem Screenshot sehen Sie **cdisk** mit einer leeren Festplatte, bereit zum Partitionieren.



cdisk mit leerer Festplatte

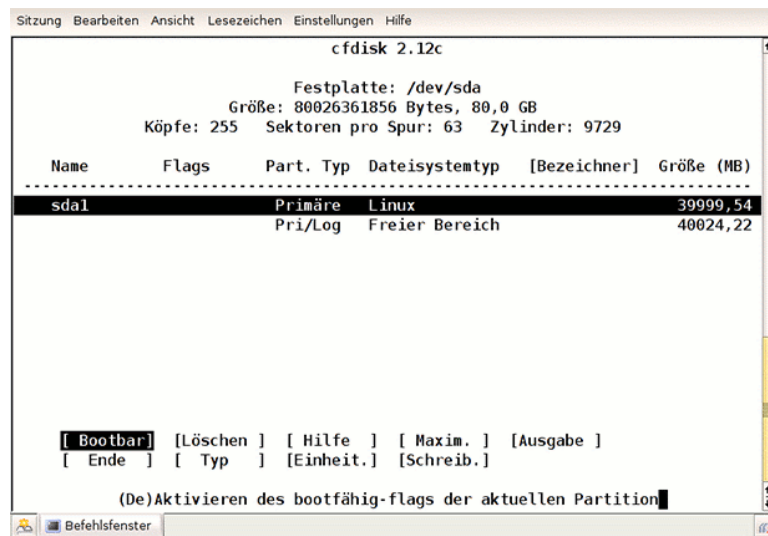
Der obere Teil des Bildschirms zeigt die Daten der Festplatte, der mittlere Teil die noch nicht vorhandenen Partitionen und der untere Teil die Auswahlmöglichkeiten von **cdisk** an.

Im nächsten Schritt gehen wir auf **Neue** und erstellen eine primäre Partition mit 40 Gigabyte.



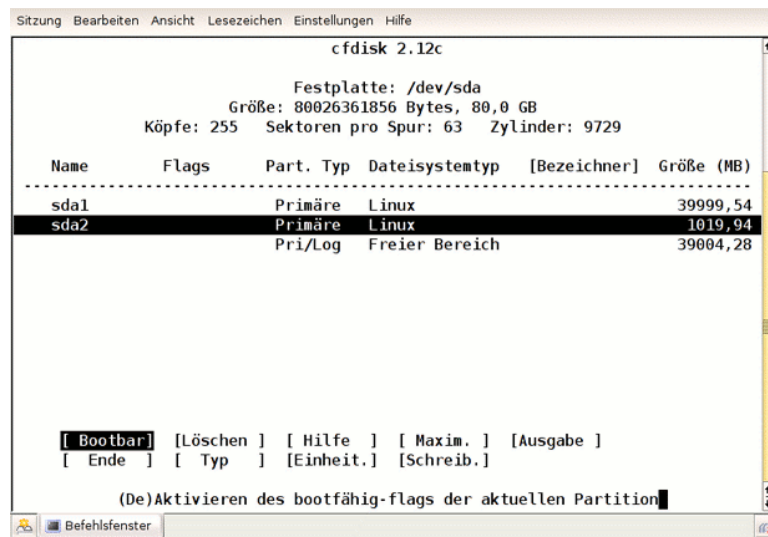
Erstellen einer primären Partition mit 40 Gigabyte

Nach dem Bestätigen mit der **Enter**taste sehen Sie Ihre Partition nun im mittleren Bildschirmabschnitt.



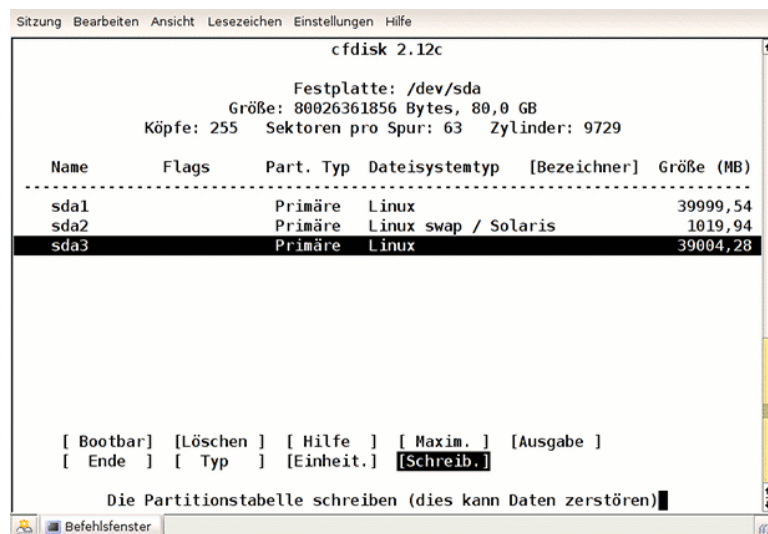
Eingetragene 40 Gigabyte

Mit der Pfeil nach unten Taste kommen Sie wieder auf den unpartitionierten Bereich der Festplatte. Im nächsten Schritt erstellen wir eine 1 GB grosse Partition, die als **swap** Speicher genutzt werden soll.



swap-Speicher

Nun gehen wir auf **Typ** und wählen Typ **82 Linux swap** aus. Den Rest der Festplatte wird wieder wie weiter oben beschrieben als primäre Partition verfügbar gemacht. Die Bildschirmausgabe sollte nun wie folgt aussehen, bevor Sie mit dem Schreiben der Partitionierungstabelle die Partitionierung übernehmen.



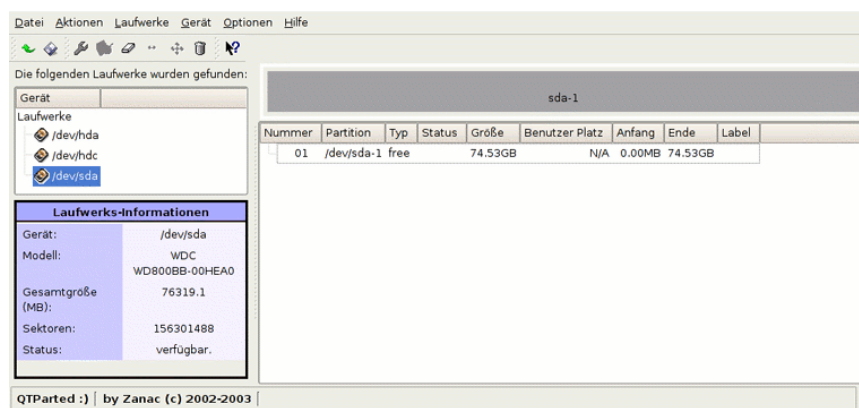
Geamtliste

In diesem Beispiel haben wir nun eine /(Root/Boot) eine **swap** und eine /home Partition angelegt, so wie sie von den meisten GNU/LINUX Installatinen unterstützt werden. Um genau zu sein:

/dev/sda1 = Root/Boot Partition 40GB  
 /dev/sda2 = swap Partition 1GB  
 /dev/sda3 = Home Partition 39GB

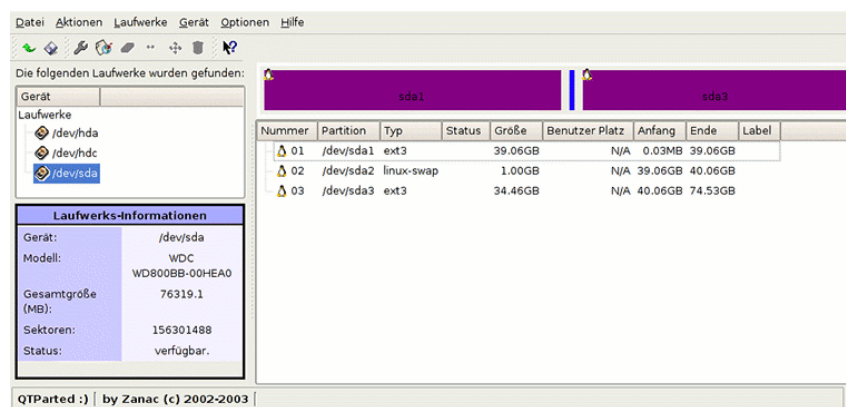
## 2.2 Partitionieren mit **qtparted**

**Qtparted** hat den Vorteil, dass es zusätzlich noch bestehende **FAT**, **FAT32** und **NTFS** Partitionen bearbeiten kann, dies machen Sie natürlich auf eigene Gefahr. Bevor Sie mit **qtparted** Windows Partitionen bearbeiten können, sollten Sie **defrag** ausführen. Ein **Backup** ist bei solchen Dateibewegungen immer anzuraten. Der Autor dieses Textes hat keine der oben genannten Dateisysteme und kann deshalb nicht auf die genannten Fähigkeiten von **qtparted** eingehen. Nun wird hier auch wieder nach dem obigen in Kapitel [Partitionieren mit cfdisk](#) genannten Schema das Partitionieren mit **qtparted** vorgenommen.



qtparted mit leerer Festplatte

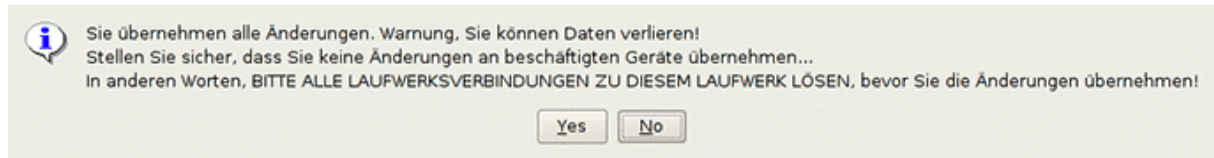
Wie Sie sehen ist **qtparted** eine in [qt](#) geschriebene grafische Anwendung. Die Aufteilung der Oberfläche ist ein wenig anders als in **cfdisk** aber deshalb nicht weniger übersichtlich. Im linken oberen Fenster unter der Menüleiste sehen Sie die im Rechner installierten Festplatten, darunter Informationen über die ausgewählte Festplatte. Im rechten Bereich des Fensters den Arbeitsbereich. Durch Anklicken der Festplatte im rechten Fenster können sie nun Partitionen anlegen.



Dies ist die gleiche Partitionsweise wie bei dem **cfdisk** Beispiel.

Über **Datei-Durchführen** übernehmen Sie die gemachten Einstellungen. **qtparted** formatiert in diesem Schritt auch das eingetragene Dateisystem. Die möglichen Dateisysteme werden im nächsten [Kapitel](#)

vorgestellt.



Sicherheitswarnung

Beachten Sie hier auch wieder die Warnung bei der Übernahme der Änderungen!

## 3 Formatieren der Festplatte

Um mit den erstellten Partitionen arbeiten zu können, müssen Sie die Festplatte noch mit einem Dateisystem versehen. Bei einem GNU/Linux System können Sie folgende Dateisysteme anlegen:

- \* **ext2**

ist das erprobte und wahre Linux Dateisystem. Es unterstützt aber kein Journaling, was bedeutet, dass routinemäßige Überprüfungen des Dateisystems beim Starten sehr zeitaufwändig sein können. Es gibt mittlerweile eine Auswahl an Dateisystemen mit Journaling neuerer Generation, die die Konsistenzchecks sehr schnell erledigen und dadurch im Vergleich den Dateisystemen ohne Journal vorzuziehen sind. Dateisysteme mit Journaling verhindern lange Verzögerungen beim Starten, wenn sich das Dateisystem in einem inkonsistenten Zustand befindet.

- \* **ext3**

ist die um Journaling erweiterte Version des **ext2** Dateisystems, das Metadata Journaling für schnelle Wiederherstellung sowie andere verbesserte Journaling Modi wie **full data** und **ordered data** Journaling unterstützt. **ext3** ist ein sehr gutes und verlässliches Dateisystem. Kurz gesagt: **ext3** ist ein universelles Dateisystem für den alltäglichen Einsatz.

- \* **ReiserFS**

ist ein B\*-tree basierendes Dateisystem mit einer guten Performance und überholt sowohl **ext2** und **ext3** im Umgang mit kleinen Dateien (Dateien kleiner als 4k) oftmals mit einem Faktor von 10-15. ReiserFS skaliert extrem gut und hat Metadata Journaling. Seit Kernel 2.4.18 ist **ReiserFS** im stabilen Kernel enthalten und sowohl als Dateisystem für generelle Anwendungen, als auch für extreme Fälle wie große Dateisysteme, den Gebrauch mit vielen kleinen Dateien und den Gebrauch mit sehr großen Dateien und Verzeichnissen mit tausenden von Dateien brauchbar.

- \* **Reiser4**

ist die neueste Variante des **ReiserFS**, es arbeitet besser mit großen Dateien zusammen als **ReiserFS** und hat automatische Dateisystem Modifikation. Es arbeitet auch mit log Dateien zusammen, die sich an verschiedenen Orten im Dateisystem befinden oder deren Ort sich ändert.

- \* **XFS** ist ein Dateisystem mit **metadata journaling**, das mit einem robusten Funktionsumfang kommt und auf Skalierbarkeit ausgelegt ist.

- \* **JFS** ist IBMs Hochleistungs Journaling Dateisystem. Es ist vor kurzem einsatzbereit geworden. Es gibt allerdings noch keine ausreichende Erfahrungen, so dass seine allgemeine Stabilität an diesem Punkt weder positiv noch negativ kommentiert werden kann.

### 3.1 Erstellen eines ext2 Dateisystems

Der Aufruf zum Formatieren lautet für **/dev/hda1**

```
root@linux / # mkfs.ext2 /dev/hda1
```

### 3.2 Erstellen eines ext3 Dateisystems

Der Aufruf zum Formatieren lautet für **/dev/hda1**

```
root@linux / # mkfs.ext3 /dev/hda1
```

oder

```
root@linux / # mkfs.ext2 -j /dev/hda1
```

Der Parameter -j schaltet das Journaling ein.

### 3.3 Erstellen eines ReiserFS Dateisystems

Der Aufruf zum Formatieren lautet für `/dev/hda1`

```
root@linux / # mkfs.reiserfs /dev/hda1
```

### 3.4 Erstellen eines Reiser4 Dateisystems

Der Aufruf zum Formatieren lautet für `/dev/hda1`

```
root@linux / # mkfs.reiser4 /dev/hda1
```

### 3.5 Erstellen eines XFS Dateisystems

Der Aufruf zum Formatieren lautet für `/dev/hda1`

```
root@linux / # mkfs.xfs /dev/hda1
```

### 3.6 Erstellen eines JFS Dateisystems

Der Aufruf zum Formatieren lautet für `/dev/hda1`

```
root@linux / # mkfs.jfs /dev/hda1
```

### 3.7 Erstellen eines swap Bereichs

Der Aufruf zum Formatieren lautet für `/dev/hda2`

```
root@linux / # mkswap /dev/hda2
```

Der Aufruf zum Einbinden der swap Partition lautet für `/dev/hda2`

```
root@linux / # swapon /dev/hda2
```