



# Der Logical Volume Manager unter Linux

## 1 Inhalt

Der Logical Volume Manager unter Linux.....	1
1 Inhalt .....	1
2 Einleitung .....	2
3 Geschichte des LVM .....	2
4 Wie funktioniert der LVM?.....	3
5 Eigenschaften des LVM.....	5
6 Praktisches Vorgehen bei der Erstellung von LV's .....	7
6.1 Erstellung eines einfachen LVM-Systems mittels CLI (Command Line Interface)8	
6.1.1 Praxisbeispiel 1: Erstellung eines 10 GB großen LVs auf einer 40 GB IDE-Platte. ....	9
6.2 Manipulieren von VGs, LVs oder PVs mittels CLI (Command Line Interface) ..	10
6.2.1 Praxisbeispiel 2: Vergrößern des soeben angelegten LVs. ....	10
6.2.2 Ergänzung: Verkleinern eines LVs.....	11
6.2.3 Praxisbeispiel 3: Vergrößern einer VG.....	11
6.3 Die graphischen Tools: .....	11
6.3.1 ASCII User Interface (UI): YaST1.....	12
6.3.2 Graphical User Interface (GUI) YaST2.....	14
7 Administration des LVM-Systems .....	16
7.1 Konfigurationsdateien.....	16
7.2 Erweiterte Befehle .....	16
7.3 Einrichten des LVM permanent im System.....	18
7.4 Verlagern von Daten auf andere PVs.....	18
7.5 Erstellen eines Snapshot-LVs zur Datensicherung.....	18
7.6 Weitere Praxisbeispiele.....	19
8 Fazit: LVM in der Praxis.....	20



## 2 Einleitung

Stellen Sie sich folgende Situation vor: Auf Ihrem Fileserver ist mal wieder der Plattenplatz zu klein geworden aufgrund der immer weiter wachsenden Datenmenge. Nachdem bis jetzt durch ökonomischeres Ausnutzen des vorhandenen Speicherplatzes das bestehende System noch ausreichte, ist es nun so weit: Neue Festplatten müssen angeschafft werden. Das bedeutet dann allerdings auch, dass sich zum einen die bisherigen Partitionen für die Verzeichnisse ändern, zum anderen Daten komplett auf die neuen Platten kopiert werden. Doch wie das bewerkstelligen, ohne die Benutzer an eine neue Verzeichnisstruktur gewöhnen zu müssen? Unter Linux kein Problem: Es werden einfach die neuen Partitionen an der gleichen Stelle des Dateibaums gemountet, so dass der User gar nichts davon merkt, dass er auf einer anderen Festplatte arbeitet. Es gibt ja keine Laufwerksbuchstaben wie unter Windows. ☺

So weit so gut. Allerdings sind Sie als Administrator eine ganze Weile damit beschäftigt und das gesamte System ist außer Betrieb. Oder es geht mal wieder ein Wochenende dabei drauf. Bei einer großen Firma ist solch ein Vorgehen außerdem aufgrund der Menge der Daten praktisch undenkbar. Wie wäre es denn, wenn Sie die neue Hardware einfach einbauen und dann nur mal kurz die vorhandenen Partitionen unmounten um sie zu vergrößern - auch über die Festplattengrenzen hinweg -, oder gar im laufenden Betrieb ihre neuen Platten integrieren? Das bietet der Logical Volume Manager unter Linux. Damit geht das freie Betriebssystem einen weiteren Schritt in Richtung UNIX als „erwachsene“ Server-Plattform.

## 3 Geschichte des LVM

Linux unterstützt seit Kernel 2.4 den Logical Volume Manager, oder kurz LVM genannt. Der LVM ist als Kernelmodul integriert und somit distributionsunabhängig. Er wird allerdings vor allem von SuSE u. a. durch graphische Tools in YaST 1 und 2 unterstützt. Ursprünglich wurde er für die UNIX-Welt von IBM entwickelt. Kommerzielle UNIX-Versionen gibt es schon länger z. B. von HP, IBM, Sun, Veritas. Im Februar 1997 entschloss sich Heinz Mauelshagen von Sistina Software, Inc. ihn für Linux zu programmieren. Seit August 2001 gibt es nun die Version 1.0 für Linux, die unter der GNU General Public License steht [1] und somit frei verfügbar ist. Sie ist stark an die HP-UX-Version angelehnt. Die Version 2.0 mit erweiterter Funktionalität ist in der Betaphase.



## 4 Wie funktioniert der LVM?

Zur Durchführung der oben erwähnten Funktionen bildet der LVM drei zusätzliche Ebenen zwischen der Partition und dem Betriebssystem ab: *Physical Volumes*, *Volume Groups* und *Logical Volumes*. Die Erklärung dieser Begrifflichkeiten erleichtert wesentlich das Verständnis für das Prinzip des LVM. Sie werden auch in *Bild 1* deutlich gemacht.

**PV (Physical Volume):** Partition, Festplatte oder logisches RAID-Drive.

**VG (Volume Group):** Container für ein oder mehrere Logical Volumes.

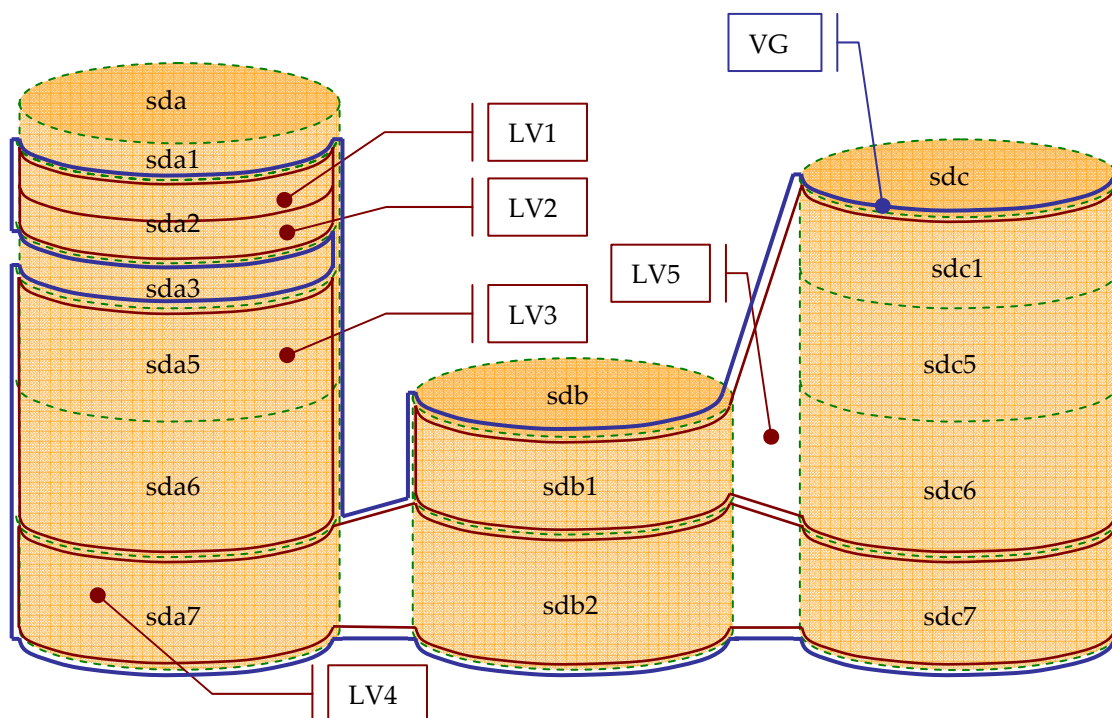
**LV (Logical Volume):** Virtuelle Partition (wird wie ein Device behandelt), Teil einer VG.

**PE (Physical Extent):** Kleinste adressierbare Einheit, per default 4 MB.

**LE (Logical Extent):** Logische Abbildung des PE im LV.

**VGDA (Volume Group Descriptor Area):** Beinhaltet die Metadaten der LVM-Konfiguration.

Auf diese Begriffe wird nachfolgend noch genauer eingegangen.



**Bild 1: Darstellung der Logical Volumes**

Eine *Volume Group* ist demnach eine erweiterbare Hülle für ein oder mehrere *Logical Volumes*, die sich unabhängig von der Hardware über mehrere *Physical Volumes* erstrecken können. Wie aus der Zeichnung ersichtlich ist, können die LVs also auch über Festplatten-grenzen hinweg verwaltet werden. Es können Partitionen sowohl in mehrere LVs unterteilt als auch mehrere in einem vereint werden.



Bei einer Datenanfrage des Kernels an die Festplatte und damit an die Device-Schnittstelle wird u. a. eine so genannte *Make-Request*-Funktion aufgerufen und die Daten dann zwecks einer Sortierung zur optimierten Abarbeitung der Sektoren für extrem kurze Zeit in einer Warteschlange gehalten. Die Daten im Platten-Cache, den der Kernel aus Performancegründen in einer dynamischen Größe hält, enthalten unter anderem eine Zieladresse in Form von physikalischen (realen) Sektoradressen des Gerätes sowie den Namen des Block-devices selbst [2]. Die Hauptaufgabe des LVM besteht jetzt darin, die realen Adressanforderungen anzunehmen und diese dann in für das logische Konstrukt LV verständliche Sektorangaben umzuwandeln, da dem System ja eine andere Partitionierung, die mit den physikalischen Partitionen nichts zu tun hat, vorgegaukelt wird. Dieses Remapping geschieht durch den Aufruf einer eigenen *Make-Request*-Funktion. Die Zuordnung der physikalischen Adressen wird hierbei durch eine logische Zwischenschicht vorgenommen.

Dazu bedient sich der LVM der Zuweisung von *Logical Extents* zu *Physical Extents* (siehe Bild 2). Wie das Bild verdeutlicht, ist sowohl das LV als auch das PV in diese Einheiten unterteilt. Auf der physikalischen Ebene sind das Zusammenfassungen von hintereinander liegenden Sektoren zu in einem PV immer gleichen, per Default und somit in der Regel 4 MB großen Einheiten. Diese *Physical Extents* haben auf der Seite des LVs als Äquivalent die *Logical Extents* der gleichen Größe. Der Unterschied liegt in der Eindeutigkeit und damit der Art der Nummerierung der Extents: während die PEs in Abhängigkeit vom physikalischen Gerät (Partition) von Anfang bis Ende derselben gezählt werden, erfolgt die Nummerierung der LEs für jedes LV bei Null beginnend. Diese Zwischenebene ist vonnöten, da eine Tabelle am Anfang jedes LVs die Adressen für die Zuordnungen bestimmt. Würde diese direkt auf die jeweiligen Sektoren verweisen, wäre sie viel zu groß.

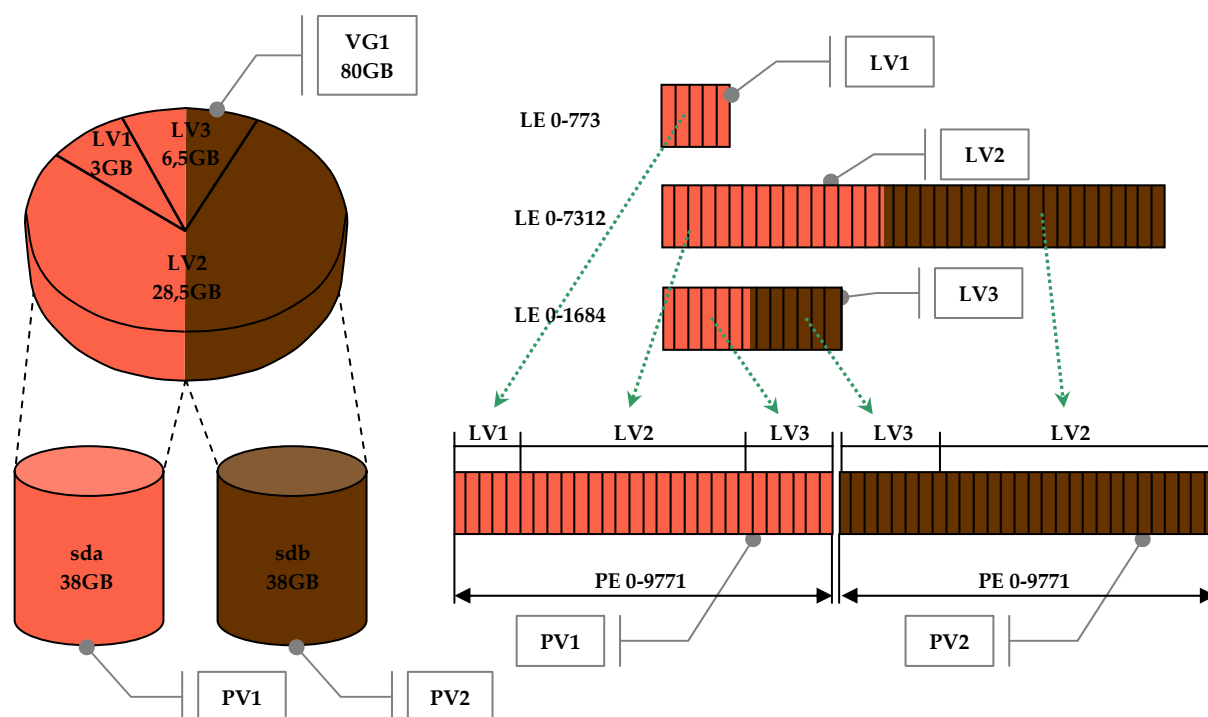


Bild 2: Zuordnung von Logical Extents zu Physical Extents (nach [3])



Die Anzahl der *Physical Extents* wird nach folgender Formel berechnet:

$$\frac{PV - \text{Größe}[MB]}{PE - \text{Größe}[MB]}$$

Analog ergibt sich die Anzahl der *Logical Extents*:

$$\frac{LV - \text{Größe}[MB]}{LE - \text{Größe}[MB]}$$

Somit ergibt sich in *Bild 2* zum Beispiel für die PEs:

$$\frac{39.088MB}{4MB} = 9.772PEs$$

Weiterhin kann man aus diesen Bedingungen auch die maximale Größe für ein LV berechnen. Da maximal 65.536 LEs in einem LV gespeichert werden können ergibt sich für 4 MB große PEs bzw. LEs nach der Formel

$$65.536 \cdot PE - \text{Größe} = 65.536 \cdot 4MB = \underline{\underline{256GB}}$$

Größere LVs erreicht man durch Erzeugen von LE/PEs mit mehr als 4 MB. Durch den Befehl

**vgcreate -s** PE-Größe ist das möglich, doch dazu später. Der Linux-Kernel 2.4 limitiert allerdings die LV-Größe auf maximal 2 TB (Terabyte).

Wie wir noch sehen werden, ist für verschiedene Funktionen das Konzept der *Volume Group Descriptor Area* wichtig. Am Anfang jedes PV sind in der VGDA folgende Metadaten gespeichert:

- Die Beschreibung des PVs
- Die Beschreibung der VG auf dem PV
- Die Beschreibungen der LVs auf dem PV
- Entsprechend der Anzahl an PEs deren Deskriptoren

Die LE-Deskriptoren werden von denen der PEs bei der Aktivierung des LVM erzeugt. Die VGDA's werden im Verzeichnis `/etc/lvmconf/` gesichert.

## 5 Eigenschaften des LVM

**Zusammenfassen von Partitionen:** Es ist möglich, aufgrund der Abstraktion von physikalischen und dem für das Betriebssystem adressierbaren Speicher mehrere Partitionen und/oder Platten in einem LV oder einer VG zusammenzufassen. Das wird in *Bild 1* dargestellt, Beispiele unter *Kapitel 6.2.1* und *6.2.3*.

**Vergrößern des Speicherplatzes:** Auch das kann bei Verwendung des entsprechenden Dateisystems (ReiserFS) online erfolgen, bei Ext2 muss die Partition nur kurz ungemountet werden (siehe *Kapitel 6.2.1, 6.2.3*).



**Verlagern von Speicher im laufenden Betrieb:** Ein großer Schritt hin zu Hochverfügbarkeitssystemen ist die Möglichkeit, mit dem LVM auch während Benutzer am System arbeiten das Speichersystem, sei es intern oder extern, sogar komplett auszutauschen (siehe *Kapitel 7.4*). Das geht doch weit beispielsweise über ein Hot-Swap RAID-System hinaus, wo nur eine (defekte) Speichereinheit erneuert werden kann.

**Erstellen von Snapshots um im laufenden Betrieb ein Backup zu fahren:** Dabei wird ein *Snapshot Device* erzeugt, das einer Momentaufnahme eines LVs entspricht. Auf diesen Alias eines vorhandenen LVs kann nur read-only zugegriffen werden, sinnvollerweise vor allem um eine Datensicherung während des Produktivbetriebes durchzuführen. Das funktioniert unabhängig von jeglicher Software, da es im Abstraction-Layer des LVM und damit auf der Dateisystemebene vor sich geht.

Tatsächlich werden dabei die sich ab dem Zeitpunkt der Snapshot-Erstellung verändernden Daten in einem speziellen Bereich mit eingeschränkter Größe schreib- und lesbar gehalten, so dass der Snapshot nach Erledigung des Backups sofort wieder gelöscht werden sollte. (Beispiel siehe *Kapitel 7.5*)

**Striping:** Ähnlich wie bei einem RAID-Stripeset gibt es beim LVM die Möglichkeit, die Daten der verschiedenen Partitionen nicht nur, wie in *Bild 1* dargestellt, linear zu mappen sondern auch „streifenweise“, wobei die LEs über mehrere PVs in festgelegten Block- oder Streifengrößen verteilt werden. Das ergäbe z. B. bei drei PVs folgende Zuordnungen:

LE1 → PV1/PE1	LE4 → PV1/PE2	
LE2 → PV2/PE1	LE5 → PV2/PE2	
LE3 → PV3/PE1	LE6 → PV3/PE2	usw.

Nachteil: eine Vergrößerung um zusätzliche PVs ist nicht möglich. Der LVM unterstützt also sowohl Software- als auch Hardware-RAID, und kann RAID 0 sogar selbst abbilden (siehe *Kapitel 7.6* sowie [5]). Empfehlenswert ist auf jeden Fall der zusätzliche Einsatz eines Hardware RAID 5-Systems vor allem aus Gründen der Datensicherheit.

**Shared Storage Clusters:** Das System scannt beim Start alle angeschlossenen Geräte wie interne Platten, externe Speichersysteme etc. auf vorhandene PVs ab. Die eigentlichen Metadaten sind ja wie gesagt am Anfang jedes PVs selbst gespeichert, von dort wird also die Konfiguration des LVM-Systems eingelesen. Im Falle von gemeinsam genutztem Speicher, so genannten Speicher-Clustern, bedeutet das also, dass der LVM nur von einem Zugangsknoten des Clusters aus aufgesetzt werden muss, alle anderen Knoten erkennen ihn automatisch. Es bedeutet allerdings auch, dass der LVM nur von einem Knoten aus administriert werden **darf**, da Veränderungen am LVM-System von den anderen Knoten nicht automatisch eingelesen werden. Daher muss bei allen anderen Knoten LVM abgeschaltet (`vgchange -a n`) und Änderungen am System nur von einem Knoten aus vorgenommen werden. Daher **Vorsicht**, das sollten Sie nur tun, wenn Sie genau wissen, was Sie tun!

**Geschwindigkeit eines LVM-Systems in der Praxis:** Theoretisch bedeutet der Mehraufwand der LVM-Verwaltung einen Geschwindigkeitsverlust des Dateisystems, da bei jeder Anfrage nach Daten deren Zuordnung zum physikalischer Speicherort in der LV-Tabelle nachgesehen werden muss. Weil das aber zum einen ein grundlegender Vorgang in einem



Rechner ist, der kaum CPU-Zeit kostet und weil weiterhin dieser relativ kleine Bereich im Cache des Linux Kernels und somit im RAM des Rechners liegt, der um mehr als den Faktor  $10^5$  schneller ist als die mechanische Festplatte, ergeben sich in der Praxis kaum Unterschiede zum normalen Dateisystem (ca. 0-3 %).

## 6 Praktisches Vorgehen bei der Erstellung von LVs

Das Erstellen eines Logical Volume Systems erfolgt in 3 Schritten:

1. Definieren und **initialisieren** des **Physical Volumes**.
2. **Definieren** der **Volume Group**.
3. **Logical Volumes** in jeder Volume Group **erzeugen**.

In Schritt 1 wählen Sie die benötigten Festplatten aus, dann gruppieren Sie diese für die Erstellung der VG zu den gewünschten Einheiten, und in Step 3 schließlich erzeugen Sie die eigentlichen LVs. Im Folgenden werden wir die Schritte zuerst per Kommandozeilen-Befehlen durchgehen und daraufhin die graphischen Tools vorstellen.



## 6.1 Erstellung eines einfachen LVM-Systems mittels CLI (Command Line Interface)

Die ursprüngliche Benutzerschnittstelle des LVM ist als CLI (Command Line Interface) implementiert und besteht aus 35 Kommandos. Die Befehlssyntax ist logisch aufgebaut und von daher einfach nachvollziehbar. Die jeweiligen Kommandos geben an *womit* man *was* macht. Ersteres ist analog zu den drei Schritten oben aufgebaut: Die Kommandos beginnen mit „**pv**“, „**vg**“ oder „**lv**“, je nachdem was manipuliert oder erzeugt werden soll. Diese Präfixe werden dann mit den entsprechenden Tätigkeiten kombiniert. Es wird auf den jeweiligen Ebenen der Speicherarchitektur etwas angelegt (**create**), gelöscht (**remove**), angezeigt (**display**) vergrößert (**extend**), verkleinert (**reduce**) umbenannt (**rename**) gesucht (**scan**) oder Attribute verändert (**change**). Daraus ergeben sich die folgenden grundlegenden LVM-Befehle sowie erweiterte zur Administration (siehe *Tabelle 4*):

Grundlegende LVM-Befehle	
Für Physical Volumes (PVs)	
<b>pvcreate</b>	Erzeugen eines PV. Ziel: Partition (Typ 8e), Festplatte, RAID
Für Volume Groups (VGs)	
<b>vgcreate</b>	Erzeugen einer neuen VG
<b>vgremove</b>	Löschen einer leeren VG ohne LV
<b>vgextend</b>	Vergrößern einer VG durch hinzufügen von LVs
<b>vgreduce</b>	Verkleinern einer VG durch entfernen von leeren LVs
<b>vgrename</b>	Umbenennen einer VG
Für Logical Volumes (LVs)	
<b>lvcreate</b>	Erzeugen eines LV in einer VG
<b>lvremove</b>	Löschen von inaktiven LVs
<b>lvextend</b>	Vergrößern eines LV durch zuweisen unbenutzten Speichers in der VG
<b>lvreduce</b>	Verkleinern eines LV (Achtung, Gefahr von Datenverlust!)
<b>lvrename</b>	Umbenennen eines LV

Tabelle 1



Mit diesen Befehlen hat man alle Werkzeuge zur Hand, um VGs, PVs und LVs zu erstellen, verändern und zu löschen. Fast alle, denn zur Größenänderung werden noch folgende Systembefehle benötigt:

Filesystem-Befehle	
<b>resize2fs</b>	Ext2 tool: Vergrößern/verkleinern eines Ext2-Dateisystems, das ungemountet sein muss.
<b>resize_reiserfs</b>	ReiserFS tool: Vergrößern eines gemounteten, verkleinern eines ungemounteten ReiserFS-Dateisystems.
<b>e2fsadm</b>	LVM/Ext2 tool: Größenänderung eines LV mit ext2-Dateisystem. Vereint und benutzt die Kommandos lvextend/lvreduce, e2fsck und resize2fs, d. h. Filesystem muss ungemountet sein.

Tabelle 2

Analog zu den oben genannten drei Schritten werden wir jetzt ein Logical Volume einrichten. [4]

### 6.1.1 Praxisbeispiel 1: Erstellung eines 10 GB großen LVs auf einer 40 GB IDE-Platte.

Wir gehen davon aus, dass die Platte als erste am 1. Controller bereits mit 3 Partitionen versehen ist:

`/dev/hda1 → 3 GB`      `/dev/hda2 → 7 GB`      `/dev/hda3 → 30 GB`

Das LV soll auf der dritten Partition eingerichtet werden.

**Partitions-ID einstellen:** Der LVM benötigt als Grundlage für seine Arbeit eine Partition des Typs „0x8e“, der extra dafür reserviert wurde. Wir erledigen das mit `fdisk`.

```
fdisk /dev/hda
```

```
Command (m for help): t            # bedeutet "Change ID"
```

```
Partition Number (1-11): 3
```

```
Hex Code (type L to list codes): 8e
```

```
Changed system type of partition 11 to 8e (Linux LVM)
```

```
Command (m for help): w            # "write table"
```

```
The partition table has been altered!
```

Vor dem Schreiben kann man noch mit `p` („print partition table“) die Änderungen kontrollieren.

**Step 1: Physical Volume einrichten.** Achtung: Durch die Deklaration der Partition als *Physical Volume* werden **alle** bisher auf der Partition befindlichen **Daten gelöscht!**

```
Syntax: pvcreate /dev/device [/dev/device]
```

```
pvcreate /dev/hda3
```



**Step 2: Volume Group einrichten.** Als wichtigster Parameter muss der Name der VG mit angegeben werden. Wir wählen hier „volgr1“. Der Device-Name darf noch nicht vorhanden sein.

```
Syntax:  vgcreate vg-name /dev/device [/dev/device]
         vgcreate volgr1 /dev/hda3
```

Die Ausführung dieses Kommandos liefert folgende Ergebnisse:

- Ein neues Device `/dev/volgr1` über dessen Dateien die VG künftig angesprochen wird.
- Eintrag der Zeile `volgr1` in die Datei `/etc/lvmtab`.
- Eine neue Binärdatei `/etc/lvmtab.d/volgr1` (s. o.).

**Step 3: Logical Volume einrichten.** Die VG stellt eine Art Hülle dar, innerhalb derer jetzt die virtuellen Partitionen angelegt werden müssen. Dazu müssen mindestens die folgenden Parameter mit übergeben werden: die *Größe* des LV (Option `-L`) und der *Name* der VG, in der es angelegt werden soll. Optional kann ein Name für das LV mit angegeben werden (Option `-n`), das System benennt die Volumes jedoch auch automatisch nach dem Schema `lvol#`. Für unser Beispiel ist uns der Name des LVs egal.

```
Syntax:  lvcreate -L size[K|M|G|T] [-n lv-name] vg-name
         lvcreate -L 10G volgr1
```

Ergebnis:

- Ein neues Device `/dev/volgr1/lvol1` das wie eine Partition angesprochen werden kann.

**Dateisystem im LV einrichten:** Das Dateisystem wird wie in einer normalen Partition mit `mke2fs` oder `mkreiserfs` angelegt. Wir legen hier ein Ext2-System an:

```
mke2fs /dev/volgr1/lvol1
```

Nun einfach noch mounten und das LV ist in das System eingebunden.

## 6.2 Manipulieren von VGs, LVs oder PVs mittels CLI (Command Line Interface)

### 6.2.1 Praxisbeispiel 2: Vergrößern des soeben angelegten LVs

Nun wollen wir auch einmal das tun, was den eigentlichen Vorteil des LVM ausmacht. Wir gehen davon aus, dass unsere 10 GB-Partition zu klein geworden ist und vergrößern sie auf das Doppelte.

1. **Vergrößern des LV:** Die neue Größe der virtuellen Partition muss mit dem Parameter `-L` an das Device übergeben werden.

```
lvextend -L 20G /dev/volgr1/lvol1
```

2. **Vergrößern des Dateisystems:**

```
umount /mountpoint # Für den Befehl resize2fs darf das LV nicht
                    gemountet sein.
```



```
e2fsck -f /dev/volgr1/lvol1 # Checken des Dateisystems auf Fehler
resize2fs /dev/volgr1/lvol1 # Vergrößern des Dateisystems auf LV-Größe
mount /dev/volgr1/lvol1 /mountpoint # Das LV wird wieder gemountet.
```

Vereinfachend können diese Schritte auch automatisch mit dem Kommando `e2fsadm` durchgeführt werden:

```
umount /mountpoint
e2fsadm -L 20G /dev/volgr1/lvol1
mount /dev/volgr1/lvol1 /mountpoint
```

### 6.2.2 Ergänzung: Verkleinern eines LVs

Bei der Verkleinerung muss beachtet werden, dass zuerst mit `resize2fs` (oder `resize_reiserfs`) das Dateisystem verkleinert wird, dann erst mit `lvreduce` das LV. Bei `resize2fs` ist die neue Größe in Blöcken anzugeben; die Blockgröße gibt das Kommando `dumpe2fs` aus.

### 6.2.3 Praxisbeispiel 3: Vergrößern einer VG

Die 40 GB der Festplatte sind erschöpft, eine zweite wird eingebaut. Deren 1. Partition wird in unsere VG eingebunden:

1. Erzeugen des neuen PV:

```
pvcreeate /dev/hdb1
```

2. Erweitern der VG:

```
vgextend /dev/volgr1 /dev/hdb1# Einbinden der neuen Partition.
```

```
vgdisplay volgr1 # Kontrolle der Größe und des freien Platzes.
```

```
VG Size 60.4 GB
```

```
Alloc PE / Size      7256 / 28.3 GB
```

```
Free PE / Size      8206 / 32.1 GB
```

```
# Auf der neuen VG sind also von 60,4 GB noch 32,1 GB frei.
```

Für mehr Befehlsbeispiele und eine genauere Syntax sowie weitergehende Informationen z. B. über das Einbinden des LVM in den Kernel (nötig z. B. bei RedHat) ist die Lektüre von [\[5\]](#) empfehlenswert.

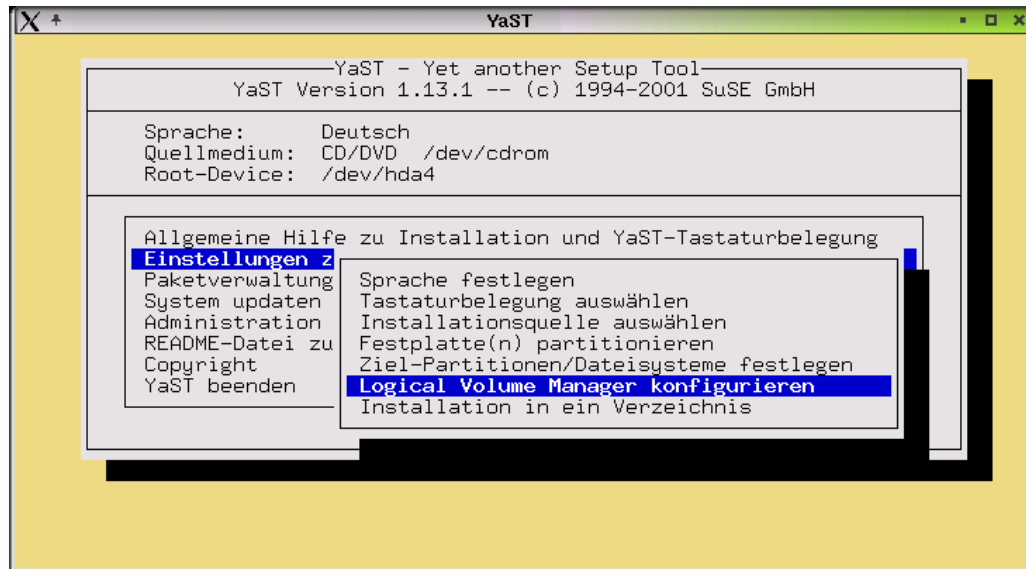
## 6.3 Die graphischen Tools:

Bei SuSE und Mandrake kann man den LVM bei der Installation auswählen, RedHat unterstützt ihn gar nicht, so dass er nachträglich installiert werden muss. Die SuSE-Distribution unterstützt als einzige den LVM zusätzlich mittels graphischer Tools. Auf eine genaue Beschreibung möchte ich im Rahmen dieses Artikels nicht eingehen, die Bedienung ist einleuchtend wenn man die Begrifflichkeiten PV, VG, LV, PE/LE verinnerlicht hat und die Wirkung der entsprechenden Kommandos kennt. Wie gehabt werden genau diese Kommandos von dem jeweiligen UI aufgerufen.



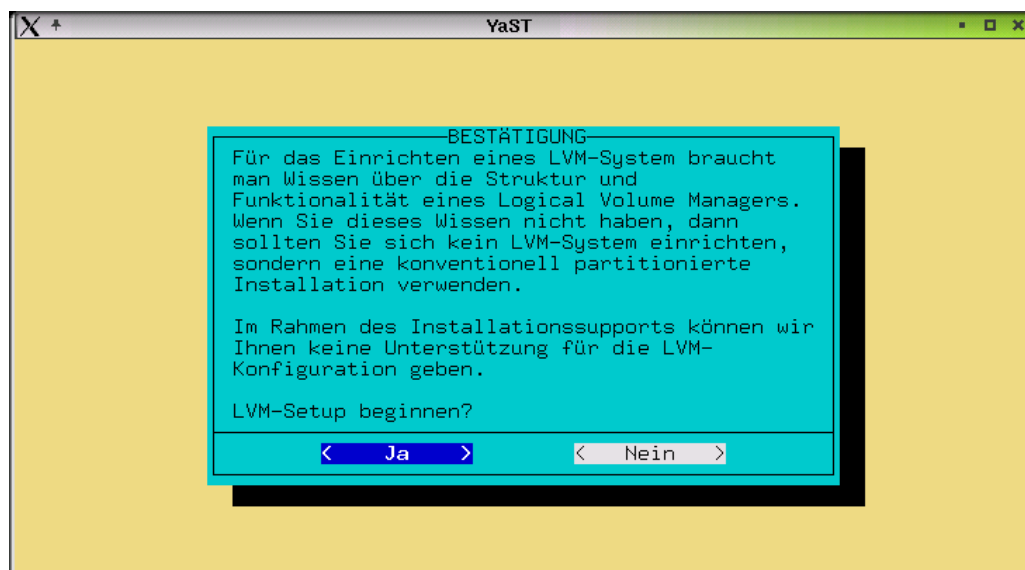
### 6.3.1 ASCII User Interface (UI): YaST1

Seit SuSE 7.0 ist der LVM über YaST1 administrierbar. Die Bedienung der strukturierten Menüs erfolgt mit der Tastatur.



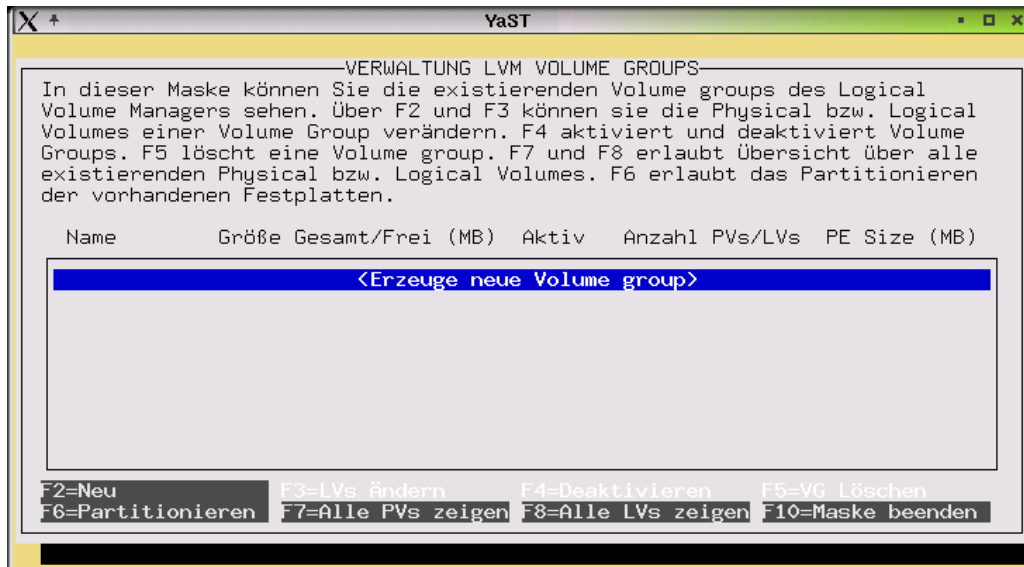
Über die Menüpunkte „Einstellungen zur Installation“ / „Logical Volume Manager konfigurieren“ gelangt man zuerst...

**Bild 3: LVM unter YaST1 starten**



...zu folgender Warnung:

**Bild 4: LVM-Warnung (YaST1)**



Auf diese Weise wird die VG erstellt. Ähnlich gestaltet sich die Administration von LVs.

**Bild 5: VG unter YaST1 erzeugen**



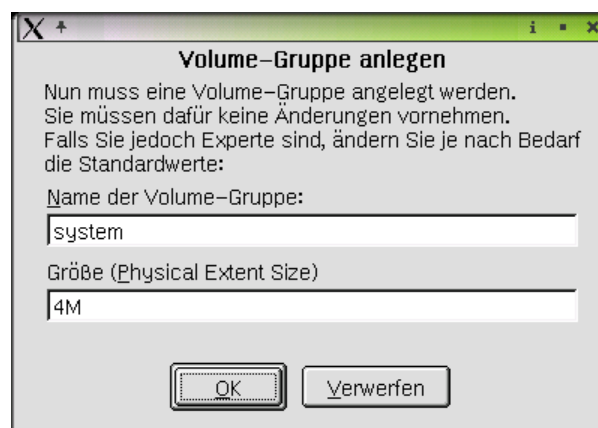
### 6.3.2 Graphical User Interface (GUI) YaST2

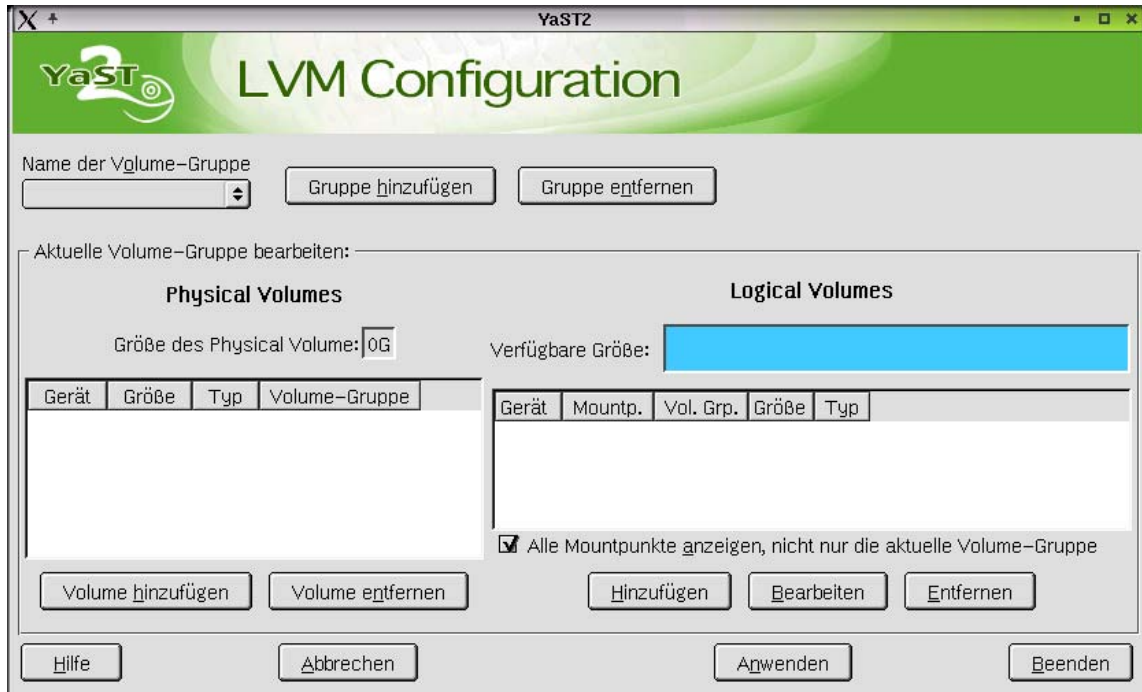
Ab der Version 7.2 steht auch ein YaST2-Tool zu Verfügung, das aber erst mit SuSE 7.3 einen praxistauglichen Reifegrad erreicht hat. Allerdings zeigte sich bei der Größenänderung eines LVs oder einer VG ein Bug, der in der `/etc/fstab` sämtliche „normalen“ Festplattenpartitionen entfernte, so dass eine Sicherungskopie dieser Datei mit anschließendem Bearbeiten per Hand erforderlich ist.



Der LVM-Manager wird unter der Rubrik „System“ ausgewählt.

Bild 6: Start des LVM in YaST2





Wenn noch keine VG angelegt ist, fordert YaST2 automatisch dazu auf.

**Bild 7: Anlegen einer VG in der LVM-Konfiguration**



## 7 Administration des LVM-Systems

### 7.1 Konfigurationsdateien

Konfigurationsdateien des LVM	
<code>/etc/lvmtab</code>	Liste mit den Namen aller VGs
<code>/etc/lvmtab.d/*</code>	Interne Beschreibung jeder einzelnen VG (enthaltene LVs, Größe etc.)
<code>/etc/lvmconf</code>	Allgemeine LVM-Konfigurationsdatei

Tabelle 3

Beim Anlegen einer VG mit dem Kommando `vgcreate` wird eine Datei `/etc/lvmtab.d/vg-name` erzeugt, in der alle für die VG relevanten Daten im Binärformat, also nicht editierbar, vorhanden sind. Sie werden von den entsprechenden Kommandos automatisch angelegt bzw. aktualisiert.

Diese sowie `/etc/lvmtab` müssen vorhanden sein. Das sind sie bei SuSE in der Regel auch, da der LVM-Dienst standardmäßig beim Systemstart initialisiert wird. Wenn nicht, rufen Sie einfach das Kommando `vgscan` auf. Sie erhalten dann zwar die Meldung „no volume groups found“, Datei und Verzeichnis werden aber leer angelegt.

In den Dateien des Verzeichnisses `/etc/lvmconf` werden unter anderem die Backups der VGDA's automatisch vom System abgelegt.

### 7.2 Erweiterte Befehle

Zur Administration sind verschiedene weitere Befehle vorhanden, mittels derer man die vorhandenen VGs und LVs kontrollieren und überblicken kann.



Erweiterte LVM-Befehle	
Allgemein	
<b>lvmdiskscan</b>	Liste aller Festplatten und Partitionen mit den darauf definierten und genutzten PVs
<b>lvchange</b>	Reset des LVM (nur wenn inaktiv!)
Für Physical Volumes (PVs)	
<b>pvdisplay</b>	Anzeigen der Attribute von PVs
<b>pvscan</b>	Liste aller PVs mit den zugeteilten VGs incl. freiem Speicher
<b>pvchange</b>	Ändern der Attribute von PVs
<b>pvdata</b>	Debugging-Anzeige der PV-Attribute aus der VGDA (s. u.)
<b>pvmove</b>	Online verlagern der LV-Daten: LEs/PEs von einem LV/PV auf ein oder mehrere andere.
Für Volume Groups (VGs)	
<b>vgdisplay</b>	Anzeigen der Attribute von VGs (physikalische und logische Volumes, deren Größe etc.)
<b>vgscan</b>	Liste aller VGs, neu erzeugen der Konfigurationsdateien
<b>vgchange</b>	Ändern von Attributen, aktivieren/deaktivieren von VGs
<b>vgexport</b>	Abmelden von inaktiven VGs zur Deinstallation und Verlagerung der zugehörigen PVs
<b>vgimport</b>	VG am Zielsystem wieder anmelden (nach pvmove)
<b>vgsplit</b>	Eine VG in zwei aufteilen, PVs in neuer VG öffnen
<b>vgmerge</b>	Zwei VGs zu einer zusammenfügen
<b>vgmknodes</b>	VG-Gerätedateien neu anlegen
Für Logical Volumes (LVs)	
<b>lvdisplay</b>	Anzeigen der Attribute von LVs (Größe, read/write Status etc.)
<b>lvscan</b>	Liste aller LVs (Größe, VG-Zugehörigkeit, aktiv/inaktiv)
<b>lvchange</b>	Ändern von Attributen, z. B. aktivieren/deaktivieren von LVs

Tabelle 4



Die Informationen der `scan-` und `display-` Kommandos werden überwiegend aus den Dateien in `/proc/lvm/` entnommen und können dort auch per Hand eingesehen werden.

Der Vollständigkeit halber seien hier noch die Befehle `vgck`, `vgcfgrestore` und `vgcfgbackup` (Daten der VGDA – Volume Group Descriptor Area – checken, sichern, wiederherstellen) sowie `lvmsadc`, `lvmsar` und `lvmcreate_initrd` erwähnt.

### 7.3 Einrichten des LVM permanent im System

Die Aktivierung des LVM geschieht mittels des Kommandos `vgchange -a y`, die Deaktivierung über `vgchange -a n`, wobei die Attribute jeweils für *activate yes/no* stehen. Sinnvollerweise geschieht dies während des Systemstarts bzw. `-stops` im Rahmen des Init-V-Prozesses für das gewünschte Runlevel. Der LVM muss vor dem Filesystem Check (`fsck`) gestartet werden, die Deaktivierung vor dem Unmounten bzw. dem Kommando `mount -no remount,ro` erfolgen.

- SuSE: automatischer Aufruf in `/etc/init.d/boot` und `/etc/init.d/halt`
- Mandrake: autom. Aufruf in `/etc/rc.d/rc.sysinit` und `/etc/rc.d/init.d/halt`
- RedHat: autom. Aufruf in `/etc/rc.d/rc.sysinit`, per Hand in `/etc/rc.d/init.d/halt!` eintragen

Weiterhin muss zum automatischen Einhängen in den Verzeichnisbaum ein Eintrag in die `/etc/fstab` erfolgen, der die LVM-Devices wie normale Partitionen einbindet:

```
/dev/volgr1/lvol1    /mountpoint    ext2 defaults 0 2
```

### 7.4 Verlagern von Daten auf andere PVs

Wie bereits erwähnt, ist es möglich, Daten im laufenden Betrieb zu verschieben, beispielsweise um Festplatten zu ersetzen. Faktisch werden dabei die PEs verschoben und dann den LEs neu zugeordnet. Die Ausführung des Befehls dauert eine Weile, ca. 4 MB pro Sekunde. Da ein Systemcrash während dieser Zeit katastrophale Folgen hätte, ist eine Datensicherung vorher ein absolutes Muss.

Beispiel: Verlagern aller Daten, die zu `lv01` gehören von einer IDE- auf eine SCSI-Platte:

```
pvmove -n lv1 /dev/hda /dev/sda
```

Der `pvmove`-Befehl ist dabei sehr flexibel. Wird kein Ziel angegeben, verteilt er, falls genügend Speicherplatz vorhanden ist, die Daten des angegebenen PV auf die anderen PVs des LV. Das ist z. B. nützlich, wenn eine Platte außer Betrieb genommen werden soll.

### 7.5 Erstellen eines Snapshot-LVs zur Datensicherung

Was ein Snapshot ist, wurde bereits in *Kapitel 5* erklärt. Die maximale Größe eines solchen beträgt das 1,1-fache des Original-LVs, d. h. während der Existenz des Snapshots dürfen maximal Veränderungen in der Größe von 10 % vorgenommen werden.



1. **Erzeugen des Snapshot-LVs:**

```
lvcreate -L Größe -s -n snap-LV-Name /dev/VG-Name/LV-Name
```

Wobei `-s` für Snapshot steht.

2. **Durchführen des Backups** mit dem Snapshot-Device.

3. **Snapshot-LV löschen:**

```
lvremove /dev/VG-Name/snap-LV-Name
```

## 7.6 Weitere Praxisbeispiele

Hier zeigen sich allerdings auch schnell die Grenzen der graphischen Tools, da viele Operationen nur mit dem CLI durchgeführt werden können. Beispielsweise würde die **Erzeugung eines Stripesets** so aussehen:

```
lvcreate -i3 -I4 -L Größe -n LV-Name VG-Name
```

Dabei bedeuten:

`-i3` → erstreckt sich über 3 Festplatten

`-I4` → Stripe-Größe = 4 KB

Für noch mehr und tiefer gehende Vorgehensweisen zur Administration und Einrichtung eines LVM-Systems möchte ich hier auf [\[5\]](#) verweisen, wo u. a. die Kommandozeilenbefehle zum Striping eines SCSI-Systems aber auch zum Hinzufügen einer neuen Festplatte, Snapshot-Backups, Verlagern und Splitting einer VG sowie die Einbindung des Wurzel-Verzeichnisses in das LVM-System aufgeführt sind. Auch auf [\[3\]](#) und das Kapitel über den Umgang mit raw I/O vor allem im Zusammenhang mit (Oracle-) Datenbanken sei hier verwiesen.



## 8 Fazit: LVM in der Praxis

Aus dem bisher Gesagten ergeben sich folgende Randbedingungen mit den entsprechenden Vor- und Nachteilen:

- Man muss sich bereits **beim Einrichten des Systems für den LVM entscheiden**, um seine Vorteile auszunutzen zu können. Für eine nachträgliche Integration in ein System muss dieses außer Betrieb genommen und die Daten vorübergehend ausgelagert werden.
- Das liegt vor allem daran, dass zu Beginn des LVM-Setups die Partitionen auf den Typ „0x8e“ neu gesetzt werden müssen.
- Ein großer Vorteil ist das **Verlagern von Daten** auf z. B. ein neues Speichersystem **während des laufenden Betriebs**.
- Bei Änderung der Größe eines LVs mit dem Ext2-Dateisystem muss dieses bei Einsatz des Standardtools `resize2fs` umgemountet werden. Bei Verwendung des nicht standardmäßig unterstützten `ext2online` ist das nicht nötig.
- Mit ReiserFS ist eine **LV-Vergrößerung** ohne Probleme auch **im laufenden Betrieb** möglich.
- Es gibt die Möglichkeit mittels eines *Snapshots* einfach ein **Backup** des Dateisystems **im laufenden Betrieb** zu erstellen.
- Der LVM ist sehr **schnell**, die Geschwindigkeitseinbußen durch den zusätzlichen Verwaltungsaufwand sind nur marginal.
- **Empfehlenswert** ist der Einsatz des LVM in Kombination mit einem **Hardware-RAID-5**, da bei Ausfall eines physikalischen Speichers sonst die Daten des gesamten LVs verloren sind. Standardmäßig unterstützt der LVM RAID-0.
- Das Root-Filesystem sollte sinnvoller weise nicht in das LV-System mit einbezogen werden. Das ist zum einen relativ kompliziert, zum anderen auch kaum nötig.

Der Logical Volume Manager ist also zu einem stabilen Tool gereift, der auch in kritischen Umgebungen – entsprechende Maßnahmen wie HW-RAID und regelmäßige Datensicherung vorausgesetzt – eingesetzt werden kann. Ob auf dem Heimrechner, dessen zweite Festplatte man einbinden will, oder der Serverumgebung mit 1.000 Usern, der LVM hat häufig seine Berechtigung und erleichtert das Administratorleben doch sehr.



---

Quellen, Infos:

- [1] Sistina: LVM-Produkte und -Support ([http://www.sistina.com/products\\_lvm.htm](http://www.sistina.com/products_lvm.htm))
- [2] Heinz Mauelshagen: Variabel teilbar (Linux-Magazin 12/01, S. 66 – 69)
- [3] Michael Hasenstein: SuSE LVM-Whitepaper  
([http://www.suse.de/en/support/oracle/docs/lvm\\_whitepaper.pdf](http://www.suse.de/en/support/oracle/docs/lvm_whitepaper.pdf))
- [4] Michael Kofler: Linux, Installation, Konfiguration, Anwendung (6. Auflage, 10/2001)
- [5] LVM HowTo von Sistina Software, Inc.  
([http://www.sistina.com/lvm\\_howtos/lvm\\_howto.pdf](http://www.sistina.com/lvm_howtos/lvm_howto.pdf))