

# Linux Grundlagen

Version 1.4, 13. Juni 2010

## Inhaltsverzeichnis

1 Vorwort.....	2
2 Die Linux Philosophie.....	3
3 Die Distributionen.....	3
3.1 Der Paketmanager.....	3
3.2 Das Betriebssystem: Der Kernel.....	4
3.3 Die Benutzerschnittstelle.....	4
3.3.1 Die Konsole: /bin/shell.....	4
3.3.2 Die grafische Oberfläche: X11.....	4
4 Einstieg in Debian.....	5
4.1 Der Debian Paketmanager: dpkg.....	5
4.1.1 Das dpkg Frontend: APT.....	5
4.1.2 Die APT Repositorys.....	6
4.2 Verzeichnisstruktur.....	6
4.3 Benutzerverwaltung: pam_unix.so.....	7
4.3.1 /etc/passwd.....	7
4.3.2 /etc/shadow.....	7
4.3.3 /etc/group.....	7
4.3.4 Dateirechte.....	8
4.4 Einbinden von Datenträgern.....	9
4.4.1 Zugriff auf Partitionen.....	9
5 Kompilieren.....	10
5.1 Vorbereitung.....	10
5.2 Kompilieren.....	10
6 Weiterführende Quellen:.....	11

# 1 Vorwort

Linux kann man nicht innerhalb von einem Tag lehren oder lernen. Linux ist nur ein Werkzeug, Um damit umzugehen muss man es Anwenden und bereit sein dazu zu lernen.

Es ist wie mit jedem Werkzeug, wenn man es ohne Bedacht und ohne Übung einsetzt, dann kann man auch viel kaputt machen.

Oft habe ich mittlerweile gehört: „Linux ist kompliziert“, „Linux ist anders“, „Linux hat gar nicht xyz“.

Stimmt! Linux ist kein Patentrezept, die Sonne scheint nicht heller, wenn ich eine Konsole öffne. Dafür schreibe ich mir meinen eigenen Programme und Treiber, wenn mein Gerät oder meine Wunschfunktion nicht unterstützt wird. Und das ganz ohne Experte auf dem Gebiet zu sein.

Es heißt die Gemeinde um Linux herum sei arrogant und unfreundlich. Nicht Arroganter als die Industrie es ist, nicht unfreundlicher als ein Supporter der schlecht bezahlt wird.

Ich persönlich distanzieren mich ausdrücklich von jeder Community! Ich gehöre keiner Distribution an und bin keinem Verhaltenscodex gebunden.

Ich helfe gerne, wenn zwei Voraussetzungen geschaffen sind:

- **Interesse**

Gerne darf mir ein Hilfesuchender sein Problem verkaufen. Ich will wissen warum ich das nun mache. Immerhin will ich mich auch selbst weiter entwickeln, und vielleicht ist er ja auf ein Problem gestoßen, das sich meiner Vorstellungskraft entzieht.

„Geht nicht“ ist KEIN Grund für mich aktiv zu werden.

- **Initiative**

Zu gerne verweise ich, bei schlecht gestellter Frage, auf die gängigen RTFM Quellen, wie Manpage, google und Dokumentation.

Mein Motto ist immer „Hilfe zur Selbsthilfe“, selbst wenn ich die Antwort kenne, warum soll ich sie direkt geben, dann fragt mich der Hilfesuchende ja immer! Es ist keineswegs verwerflich etwas nicht zu wissen, aber ebenso interessant wie die Antwort ist der Weg dort hin.

## 2 Die Linux Philosophie

Der Kernel [7] steht unter der GPL. Initial ist er von Linus Torwald entwickelt worden, stark an gelehnt an den im studentischen Umfeld verbreiteten Unix. Linux sollte nicht nur frei sein, wie es Freibier ist, sondern auch Quelloffen. Jeder hat die Möglichkeit an der Entwicklung mitzuwirken. Die einzige Bedingung ist, dass die neue Entwicklung wieder offen gelegt wird.

Jeder Privatanwender hat die Möglichkeit ein Projekt zu starten und zu entwickeln. Um so interessanter es auch für andere Entwickler ist, desto mehr die Möglichkeit zu wachsen. Siehe MySQL, OOo.

Die Schattenseite ist die nicht gegebene Garantie. Wenn ein Projekt kein Interesse mehr bindet, so kann es sterben. Eine Firma die dieses Programm nutzt muss sich neu orientieren oder auf alternativen umsteigen.

## 3 Die Distributionen

Linux selbst ist nur der **Kernel**, mit dem der normale Anwender wenig zu tun hat. Um für die Masse attraktiv zu sein fehlt noch Anwendungssoftware.

Wie der Kernel selbst ist vieles dieser Software auch unter freien Lizenzen im Internet zu finden. Nimmt man die Software, stellt diese bequem mittels **Paketmanager** zur Verfügung und ergänzt ein wenig Support und Dokumentation, so hat man eine Distribution.

Es gibt kostenpflichtige Distributionen wie *Suse* oder *Redhat Enterprise*, die neben den Paketen auch noch kostenpflichtigen Support und Handbücher anbieten. Und es gibt die sogenannten 'Community Distributionen', wie *Debian*, *openSuse* oder *Gentoo*, die größtenteils von der Community getragen werden.

### 3.1 Der Paketmanager

Ein Paketmanager sorgt dafür, das die vielen auf verschiedene Art und Weise geschriebenen und gepflegten Programme auf einen Nenner kommen und das System trotz der Vielfalt einfach zu Erweitern und zu Warten bleibt.

Ein Programm hat einen **Entwickler** der klassisch den Source geschrieben hat und nun pflegt. Dazu hat es einen **Maintainer**, der diesen Source aufbereitet und für die gewählte Distribution packt. Ein Maintainer kann mehrere Programmpakete pflegen oder auch für mehrere Distributionen packen.

Jeder Paketmanager hat Vor- und Nachteile, auch hier gilt: 'Ein Patentrezept gibt es nicht'.

Distribution	Paketmanager	Frontend
Debian, Ubuntu	dpkg	APT (apt-get, aptitude)
RedHat, SUSE	rpm	Yast, Yum
Slackware	pkgtools	
Gentoo	portage	Emerge

## 3.2 Das Betriebssystem: Der Kernel

Linux selbst ist nur der Kernel, dieser liegt nach der Installation [9] unter `/boot/vmlinuz` und wird mittels Modulen, die unter `/lib/modules/` zu finden sind, erweitert. Die unterstützten **Architekturen** sind auf [kernel.org](http://kernel.org) nachzulesen.

Der Kernel setzt direkt auf die Hardware auf, mittels der Library **glibc** werden die Grundfunktionen eines minimalen Systems zur Verfügung gestellt.

Beim **kompilieren** eines Kernels geht es um verschiedene Faktoren, wie Geschwindigkeit (z.B. im Kernel ist es schneller) und Sicherheit (z.B. ein Modul kann einen Segfault verursachen, ohne dass das System stoppt).

Mein Lieblingsbeispiel aus der Praxis ist die Netzwerkkarte. Im Kernel **monolithisch** kompiliert ist sie zur Bootzeit verfügbar und es kann direkt drauf zugegriffen werden. Wenn allerdings von außen ein Pingflood auf die Netzwerkkarte einwirkt, dann würde sie den Kernel mit belasten. Als Modul kann der Kernel sie mit entsprechenden Hilfsprogrammen bei Bedarf entladen.

## 3.3 Die Benutzerschnittstelle

### 3.3.1 Die Konsole: `/bin/shell`

Der Benutzer findet sich unter Linux immer wieder auf der Shell. Was für den ambitionierten Heimanwender, der Windows oder MacOSX gewohnt ist, als rückständig und kompliziert erscheint, ist für den Linux Administrator die Quelle der Macht. Ein oft genannter Grundsatz der Distributionen ist „Für jede Anwendung ein Tool“, das gilt für die Konsole, wie für die Grafische Oberfläche.

Als 'echte **Konsolen**' findet man mit `[Alt][Fn]` die **ttyn**, unter X kann man sich die **pty[n]** (**Pseudoterminals**) holen.

Die Standard Shell ist zum Beispiel *bash* oder *dash*, aber im studentischen Umfeld trifft man auch auf die *csch* oder *zsh*. Erlaubt ist, was gefällt.

Um sich auf der Shell zurechtzufinden gibt es Standardbefehle. Als Einstieg empfiehlt sich als Referenz ein gutes Buch [1][2], eine Kaffeetasche oder ein Hintergrundbild [3] mit einfachen Befehlssatz.

Ein großer Vorteil der Arbeit auf der Shell sind die Umleitungen und Pipes. Erklärt auf [bin-bash.de](http://bin-bash.de) [8]

### 3.3.2 Die grafische Oberfläche: X11

Die Konsole ist aber nicht alles. Über das Protokoll **X11** gibt es unter Linux den **X-Server** (z.B. *Xorg*), der hält die Kontrolle über die Hardware, wie Maus, Tastatur und Bildschirm. Der **X-Client** sagt an, was auf dem Bildschirm ausgegeben werden soll.

Damit es gut aussieht wird ein **Windowmanager** (z.B. *XFCE*, *Fluxbox*, *e17*) oder aber ein ganzes **Desktopenvironment** (*Gnome*, *KDE*) geladen.

Und wiederum, „Erlaubt ist was gefällt“, jede Distribution hat Empfehlungen, kaum etwas kann nicht individuell angepasst werden.

Das man nicht an eine Oberfläche gebunden ist ist für den Einzelplatz vielleicht ein Segen, für viele Administratoren und Entwickler aber auch die Hölle.

## 4 Einstieg in Debian

### 4.1 Der Debian Paketmanager: dpkg

Der Paketmanager für Debian ist **dpkg**. Bequem mittels der **APT** Library lässt sich dpkg bequem benutzen. Die Frontends sind z.B. *apt-get* [5], *aptitude*, *Adept*.

Während es damals einfach war, RedHat nutzte YUM, SUSE hatte YAST und Debian hatte apt-get, so sind die Systeme heute quer portiert worden.

#### 4.1.1 Das dpkg Frontend: APT

Unter Debian werden die Softwarepakete mittels dpkg installiert und entfernt, das allerdings macht eine Distribution noch nicht Benutzerfreundlich. Auf dpkg wurde die Schnittstelle APT gesetzt, auf die z.B. das Frontend **apt-get** aufsetzt. Mittels der APT Library ist es möglich Paketquellen (Repositorys) zu verwalten, Abhängigkeiten automatisch aufzulösen und relativ neu ist die Möglichkeit auch nicht mehr benötigte Pakete zu deinstallieren.

Als Beispiel wollen wir dein Programm für passives Fingerprinting installieren, wir suchen nach diesen Begriffen mittels:

```
lupus@zoe:~$ apt-cache search passive fingerprint
ettercap - Multipurpose sniffer/interceptor/logger for switched LAN
ettercap-gtk - Multipurpose sniffer/interceptor/logger for switched LAN
p0f - Passive OS fingerprinting tool
```

Und erhalten drei Pakete als Ergebnis. Da die Beschreibung schon genau enthält was wir suchen installieren wir **p0f** mittels:

```
lupus@zoe:~$ sudo apt-get install p0f
[sudo] password for lupus: *****
[...]
Entpacke p0f (aus .../p/p0f/p0f_2.0.8-2_amd64.deb) ...
[...]
Schreibe erweiterte Statusinformationen... Fertig
```

Ohne das ich etwas machen musste hat apt-get die Abhängigkeiten aufgelöst, alle benötigten Paket aus den **bekannten Quellen (sources.list)** geladen und Installiert.

Unter Umständen werden in einem Dialog noch weitere Informationen während der Installation abgefragt. Das System dahinter nennt sich 'Debconf' und passt anhand dieser Fragen gegebene Templates an und setzt diese als Konfigurationsdateien. Die Templates und Fragen werden vom Maintainer de Pakets vorgegeben.

## 4.1.2 Die APT Repositorys

Die Datei `/etc/apt/sources.list` gibt an, wo APT die Pakete findet. Der Aufbau der Quellen ist streng vorgeschrieben, als Protokoll gibt es viele Alternativen, als Beispiel: `http://`, `ftp://`, `file://`.

```
deb ftp://ftp2.de.debian.org/debian stable main contrib non-free
```

**deb**, Dateityp

**ftp://**, Protokoll

**ftp2.de.debian.org**, URL des Paketsspiegels

**/debian**, Verzeichnis auf dem Spiegel

**stable**, Angabe der Distribution

**main contrib non-free**, Angabe des Baumes (Tree)

## 4.2 Verzeichnisstruktur

Unter Linux gibt es keine Laufwerke, auf die man direkt zugreifen kann. Alles wird im Verzeichnisbaum dargestellt. Der Stamm ist `/` und darunter werden nach Bedarf weitere Datenträger als Verzeichnis eingebunden.

Der allgemeine Standard ist genauer definiert in dem „Filesystem Hierarchy Standard“ FHS [6].

`/` root, der Stamm

`/boot`, Kernel, **initrd**

`/bin`, ausführbare Dateien, die zur Bootzeit gebraucht werden

`/dev`, Devicefiles, Dateien die ein Gerät repräsentieren

`/etc`, Konfigurationsdateien und globale Vorgaben

`/home`, Heimatverzeichnis (`~`) der Benutzer, ggf. eigene Partition

`/lib`, Librarys und Shared Objects auf die Programme zurückgreifen

`/media`, Automatisches Einbinden (**udev**, **HAL**) von Datenträgern

`/mnt`, Empfehlung zum manuellen Einhängen von Datenträgern

`/opt`, Empfehlung für proprietäre Software

`/proc`, Prozessdateisystem für den Zugriff auf Systemressourcen

`/root`, Heimatverzeichnis des Superusers (root)

`/sbin`, ausführbare Dateien (root), die zur Bootzeit gebraucht werden

`/tmp`, temporäre Dateien, ggf. eigene Partition

`/usr`, Unix System Ressource, Distributionsabhängige Nutzung

`/var`, Variable Dateien, wie Spooler, Datenbanken, Logdateien

`/boot/vmlinuz` (kernel)

`/dev/hda1` (block device)

`/dev/psaux` (character device)

`/usr/share/doc/` (Dokumentationen)

`/usr/src/` (Quellcode, header)

### 4.3 Benutzerverwaltung: pam\_unix.so

Die Benutzerverwaltung wird mittels PAM (Pluggable Authentication Module) organisiert und der Installationsstandard ist das pam\_unix.so Modul. Hier werden die Benutzer in Dateien verwaltet.

Ein Benutzername oder Gruppenname ist nur ein Alias um das System für Menschen lesbar zu machen. Intern, zum Beispiel auf dem Dateisystem, wird aber NUR eine Nummer, die UID und GID abgespeichert.

#### 4.3.1 /etc/passwd

In dieser Datei stehen die Benutzer eines Systems. Unter Debian liegen die Systembenutzer unterhalb der UID 1000, und ie Benutzer fangen ab 1000 an.

Der Superuser hat immer die UID und GID 0. Als Standard ist der Name root vorgegeben, aber auch wheel ist anzutreffen.

```
lupus:x:1000:1000:Benjamin Moeller,,,:/home/lupus:/bin/bash
```

Loginname  
Passwort (x = gesetzt, ! = nicht gesetzt)  
UID  
GID  
GCOS Informationen  
Heimatverzeichnis  
Standard Shell

#### 4.3.2 /etc/shadow

Das Password wird als MD5 Hash gespeichert. Da auf weitere Informationen (GCOS Felder) jeder Zugriff haben sollte wurde das Passwort ausgelagert in eine Datei, die nicht von jedem lesbar ist.

```
lupus:$6$04A[...]fqIcwv/:14569:0:99999:7:::
```

Loginname  
Passwort als Hash  
letzte Änderung  
nächste Änderung  
Alter  
Passwort Warnung  
Fallback Akzeptanz  
Ablaufdatum  
Reserviertes Feld

#### 4.3.3 /etc/group

```
lupus:x:1000:
```

Loginname  
Gruppenpasswort  
GID  
weitere Benutzer der Gruppe

### 4.3.4 Dateirechte

Das Standarddateisystem für Debian basierende Betriebssysteme ist **ext3**. Dieses Dateisystem bietet keine Komplexen ACLs, wie *NTFS*.

Es gibt für die Dateirechte 3 Stufen. Den Besitzer, die Gruppe und alle Anderen. Jede Stufe hat 3 mögliche Berechtigungen oder hat die Berechtigung nicht. **Lesen (r), schreiben (w), ausführen (x)**.

Vorangestellt ist der Typ einer Datei. Wie erwähnt wird alles durch eine Datei repräsentiert, so kann sie z.B. den Typ Verzeichnis (d), blockorientiertes Gerät (b), symbolischer Link (l) haben.

Daraus ergibt sich eine Matrix von:

<pre>lupus@nina:~\$ ls -lh /etc/shadow -rw-r----- 1 root shadow 1,2K 2010-03-25 09:20 /etc/shadow</pre>	
(l) Dateirechte	
(l) Anzahl der Verlinkungen auf diese Datei	
(l als UID/GID) Benutzer und Bevorzugte Gruppe	
(l) Größe der Datei in Byte, durch den schalter -l in Kbyte	
(l als Timestamp inkl. Erstellungsdatum) Datum und Zeit der letzten Änderung	
(D) Dateiname	
(l = Gespeichert im Inode, D = gespeichert in der Datei)	

Es ist eine Datei (-), der Benutzer (root) darf die Datei lesen und Schreiben aber nicht ausführen (rw-), die Gruppe (shadow) darf die Datei nur lesen (r--), alle anderen dürfen gar nichts (---).

Dabei gilt immer die Zugehörigkeit mit dem höchsten Wert. Wenn der Benutzer passt, wird nicht mehr nach den Gruppenrechten geschaut.

Die Dateirechte werden klassisch als Zahlenwerte dargestellt, die addiert einen 'Berechtigungscode' ergeben. Lesen 4, schreiben 2, ausführen 1. Für das oben genannte Beispiel haben wir also das Recht 640.

Für das schnelle setzen der Rechte ohne Taschenrechner kennt 'chmod', der Befehl zum Ändern der Dateirechte, auch Buchstaben.

**Besitzer (o owner), Gruppe (g group), Alle (a all).**

Hinzufügen von Schreibrechten der Gruppe: `chmod g+w /pfad/datei`

Entfernen vom Ausführungsrecht Aller: `chmod a-x /pfad/datei`



## 4.4 Einbinden von Datenträgern

Das Mounten, einbinden von Datenträgern, unter Linux unterscheidet sich von Windows, da es keine Laufwerksbuchstaben pro Datenträger gibt, sondern alles in den Verzeichnisbaum eingebunden wird.

Manuelle bindet man Datenträger mit dem Befehl **mount** ein, wobei der **Mountpoint** existieren muss.

```
# mkdir /mnt/test  
mount -t <dateisystem> -o <optionen> /<devicefile> /<mount>/<point>  
# mount -t vfat -o defaults,users /dev/sdb1 /mnt/test
```

Der Vorgang lässt sich über die Datei /etc/fstab [4] automatisieren

### 4.4.1 Zugriff auf Partitionen

Ein Blockdevice (z.B. /dev/**sda1**) unter Linux ist dabei wie folgt aufgebaut:

**hd/sd h** sind AT-Geräte, bekannt als z.b. IDE. Das **s** steht für SCSI Geräte.

**a** steht für das erste erkannte Gerät. Unter IDE gab es den 1. und 2. primären, sowie den 1. und 2. sekundären Port, das ergab die Buchstaben **a-d**. Unter SCSI gab es klassisch 7 mögliche Geräte an einem Port, **a-g**. Das kann heute mehr sein, da SATA, CD-Brenner, USB Massenspeicher alle als SCSI Geräte angesprochen werden.

**1** ohne eine Ziffer ist das ganze Gerät gemeint, inklusive **MBR** (Master Boot Record). Die **1** steht für die erste Primäre Partition, auf diesem Gerät. Dabei können durch die geringe Größe im MBR nur *4 Primäre Partitionen* erstellt werden. Alternativ kann man bis zu *3 primäre Partitionen und einen erweiterten Container* anlegen, in dem logische Partitionen liegen. Die erweiterten Partitionen fangen ungeachtet der Anzahl der primären Partitionen bei **5** an aufwärts zu zählen.

## 5 Kompilieren

Nicht jedes Programm oder Kernelmodul liegt immer als Paket vor, dann bleibt noch immer die Möglichkeit aus dem Quelltext selbst zu kompilieren. Der einfachste Weg ist über den Dreisatz `./configure; make; make install`, vorausgesetzt alle Abhängigkeiten sind schon erfüllt.

### 5.1 Vorbereitung

Die wichtigsten Programme und Abhängigkeiten unter Debian installiert man mit folgender Zeile:

```
# aptitude install wget bzip2 gcc kernel-package linux-kernel-headers
binutils bin86 make fakeroot build-essential
```

Zuerst laden wir das Paket herunter, entweder über einen Browser oder mittels `wget`. Für mich persönlich hat es sich bewährt einen Unterordner `sys` im Benutzerverzeichnis anzulegen, in dem ich Quelltexte herunterlade und bearbeite.

```
$ cd ~/sys/
$ wget <programm-src>.tgz
$ tar xvzf <programm-src>.tgz
$ cd <programm>
```

In dem Unterordner `<programm>` findet sich hoffentlich eine Datei namens `INSTALL` oder `README`. Hier finden sich weitere Hinweise zu den Abhängigkeiten und Besonderheiten.

Ist ein Programm von einem anderen Abhängig, so benötigen wir die **Header**. Ist die Abhängigkeit als Paket vorhanden, dann wird mit `aptitude install <paket>-dev` alles notwendige installiert.

### 5.2 Kompilieren

```
$ ./configure [--help]
$ make
$ su
# make install
```

Der Befehl `./configure` besitzt in der Regel den Schalter `-help`. Wenn er vorhanden ist schadet es nicht einmal anzusehen was für Optionen es gibt.

Die meisten Programme brauchen **Rootrechte** zum installieren, nicht aber zum Kompilieren. Daher muss das Rootrecht auch nur im letzten Schritt gegeben werden.

Der Befehl `make` führt das **Makefile** im Verzeichnis aus, in dem verschiedene Anweisungen zum Kompilieren stehen.

## 6 Weiterführende Quellen:

- [1] <http://www.oreilly.de/catalog/linuxnut4ger/index.html>
- [2] <http://www.amazon.de/Linux-Michael-Kofler/dp/3827321581>
- [3] <http://www.lupuse.org/linux/img/>
- [4] <http://www.lupuse.org/linux/etc/fstab>
- [5] [http://www.lupuse.org/linux/quick\\_n\\_dirty/apt-debian](http://www.lupuse.org/linux/quick_n_dirty/apt-debian)
- [6] <http://www.pathname.com/fhs/>
- [7] <http://www.kernel.org/>
- [8] <http://bin-bash.de/komfort.php>
- [9] [http://www.lupuse.org/linux/quick\\_n\\_dirty/kernel\\_debian](http://www.lupuse.org/linux/quick_n_dirty/kernel_debian)