

# **Linux-Fortbildung**

## Teil I

# **Grundlagen**

IFB-Nr. 19.287  
14. – 16. Januar 2002  
IFB Haus Speyer

Daniel Jonietz  
([daniel@jonietz.de](mailto:daniel@jonietz.de))

## Literatur-Vorschläge

### einleitend:

- Günther, K.: **Linux Kompaktreferenz**  
MITP-Verlag, Bonn 2000.  
ISBN 3-8266-0594-2 (DM 24,80)
- Siever, E.; Spainhour, St.; Figgins, St.; Hekman, J. P.: **Linux in a nutshell**  
Deutsche Ausgabe  
O'Reilly Verlag, Köln <sup>3</sup>2001.  
ISBN 3-89721-195-5 (DM 69,-)
- Kofler, M.: **Linux**  
Addison-Wesley Verlag, München <sup>5</sup>2001.  
ISBN 3-8273-181-0 (DM 69,90)

### vertiefend:

- Newham, C.; Rosenblatt, B.: **Lerning the bash Shell**  
O'Reilly & Associates, USA <sup>2</sup>1998.  
ISBN 1-56592-347-2 (\$ 24.95)
- Krienke, R.: **UNIX Shell-Programmierung**  
Carl Hanser Verlag, München <sup>2</sup>2001.  
ISBN 3-445-21722-3 (DM ?)
- Rathmann, M.; Wieskotten C.: **Jetzt lerne ich Shell-Programmierung**  
Markt+Technik Verlag, München <sup>2</sup>2002.  
ISBN 3-8272-6241-2 (EUR 24,95)

## Vorbemerkung

Dieses Skript stellt keinen Anspruch auf Vollständigkeit — an vielen Stellen sind bewußt Leerstellen angebracht. Es soll lediglich ein Skelett für eigene Notizen darstellen und die Nachbereitung vereinfachen.

# 1 Einführung

Linux erfreut sich zweifelsohne zunehmender Beliebtheit. In diesem Kurs sollen die absoluten Grundlagen für einen fundierten Umgang mit einem Linux-System gelegt werden. Wie der Titel verspricht ist er also tatsächlich eher als ein Grundlagenkurs denn als Einführung angelegt, wobei natürlich der Tatsache Rechnung getragen werden muss, dass Anfänger ohne Einführung nicht auskommen werden. Insofern also ein einführender Grundlagenkurs.

Wir gehen im Folgenden davon aus, dass ein Linux-System bereits erfolgreich eingerichtet wurde — die Erstinstallation stellt heute dank grafisch geführter Installationsroutinen i.d.R. kein Problem mehr dar. Vorab soll aber noch geklärt werden, was Linux eigentlich ist.

## 1.1 Was ist Linux?

**Linux** ist ein UNIX-ähnliches Betriebssystem, aber genau genommen trifft der Begriff „Linux“ nur den Kernel, d.h. den innersten Teil des Betriebssystems mit elementaren Funktionen wie Speicher- und Prozessverwaltung. Eine **Distribution** ist eine Einheit aus eigentlichem Betriebssystem und Zusatzprogrammen, die i.d.R. eine einfache Installation durch eigene Tools (z.B. YaST bei SuSE) ermöglicht. Linux baut auf einer breiten Basis frei verfügbarer Software auf, wesentlich sind die GNU-Tools. Das sind unter gewissen Einschränkungen frei verfügbare Programme — und zwar mitsamt den Quellen. Software, die im Sinne von GNU entwickelt wird, unterliegt der GNU General Public License (kurz GPL) die ausschliesst, dass jemand ein GPL-Programm weiterentwickeln und verkaufen kann, ohne die Änderungen für alle öffentlich verfügbar zu machen. Durch jede Weiterentwicklung eines Programmes, das der GPL unterliegt, gewinnen folglich alle. Erst die Kombination von Linux-Kernel, GNU-Tools, Netzwerksoftware, dem X-Window-System und vielen anderen Bestandteilen ergibt sich eine vollständige Linux-Distribution mit der sich ein Linux-System als das darstellt, was unter „Linux“ verstanden wird.

## 1.2 Anmeldung

Ein Linux-System kann grundsätzlich erst nach einer Anmeldung bedient werden. Zu einer vollständigen Anmeldung gehören ein **Benutzername** (Login-Name) und ein **Passwort**. Nach der erfolgreichen Anmeldung wird der Nutzer vom System begrüßt und ihm eine Eingabeaufforderung bereitgestellt — die Arbeit kann beginnen. Nach der Arbeit sollte man sich wieder abmelden, dazu dienen die Kommandos `logout`, `exit` oder die Tastenkombination `Ctrl`+`d`.

# 2 Die Shell

Die Shell stellt eine **Schnittstelle zwischen System und Benutzer** dar und ermöglicht ihm die einfache Ausführung von Kommandos. Dabei ist die Shell kein integraler Bestandteil des Betriebssystems, sondern ein normales Programm das sogar austauschbar ist.

## Wichtige Shell-Varianten

- bsh (bourne-shell)
- bash (bourne-again-shell)
- ksh (Korn-shell)
- csh (C-shell)
- tcsh (TC-shell)

Wir werden ausschließlich die `bash` verwenden und behandeln.

## Einsatzzwecke der Shell

- Einsatz als Kommandointerpreter
- Einsatz als Programminterpreter ← Shell-Programmierung

## Prinzipieller Ablauf eines Kommandos

- Shell signalisiert Bereitschaft zum Empfang von Kommandos durch **Prompt-Zeichen**
- Benutzer gibt Kommandozeile ein
- Shell wertet die Kommandozeile aus, nimmt ggf. Ersetzungen vor, startet Kommando(s) und gibt schliesslich Ausgabe und evtl. Fehlermeldungen an den Benutzer weiter.

### 2.1 Generelle Syntax eines einfachen Kommandos

- *Befehl Option Parameter*
- wobei *Option* je nach Befehl, Version und konkreter Option entweder durch
  - ein `--`-Zeichen und einen die Option spezifizierenden Buchstaben (Kurz-Notation)oder durch
  - zwei `--`-Zeichen und ein die Option spezifizierendes Wort (Lang-Notation)dargestellt wird.
- Beispiel: Der Befehl `ls` (von list) gibt den Inhalt eines Verzeichnisses aus und kennt lange und kurze Optionen:
  - `ls -l` erzeugt eine lange, d.h. detaillierte Ausgabe
  - `ls --help` gibt einen Hilfetext aus

## 2.2 Komfort

Eine gute Shell erleichtert dem Anwender die Bedienung: In der `bash` können Namen mittels `Tab` vervollständigt werden, die bereits eingegebenen Kommandos können mittels der Pfeil-Tasten `↑` und `↓` wiederholt oder bearbeitet werden.

## 2.3 Ersetzungen

Bevor die Shell die Kommandos an das Betriebssystem zur Ausführung übergibt ersetzt sie spezielle Zeichen (sogenannte **Metazeichen**) in der Kommandozeile:

Metazeichen	Erklärung
<code>*</code>	ein oder mehrere beliebige Zeichen
<code>?</code>	genau ein beliebiges Zeichen
<code>s</code>	genau eines der Zeichen aus der Zeichenkette <code>s</code>
<code>!s</code>	genau ein beliebiges Zeichen, das nicht in <code>s</code> vorkommt
<code>[c<sub>1</sub>-c<sub>2</sub>]</code>	genau ein beliebiges Zeichen aus dem Bereich <code>c<sub>1</sub></code> bis <code>c<sub>2</sub></code>
<code>{s<sub>1</sub>, s<sub>2</sub>, ...}</code>	genau eine beliebige Zeichenkette <code>s<sub>1</sub></code> oder <code>s<sub>2</sub></code> oder ...

Die Wirkung der Metazeichen kann durch **Quoten** aufgehoben werden:

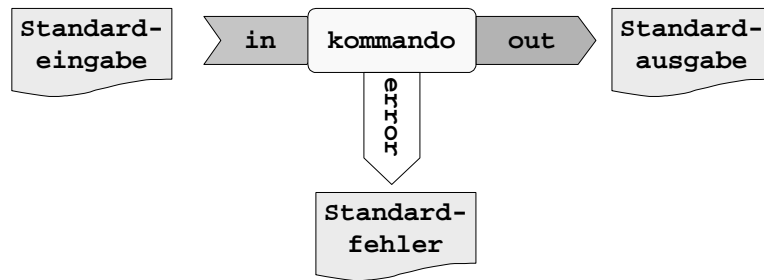
Quotezeichen	Erklärung
<code>\</code>	hebt die Sonderbedeutung des direkt rechts stehenden Zeichens auf
<code>' '</code>	der innerhalb der einfachen Anführungszeichen stehende Text wird durch die Shell nicht verändert
<code>" "</code>	wie <code>' '</code> , nur werden Variable durch ihren Wert ersetzt
<code>'kommando'</code>	Ersetzt den Text <code>kommando</code> durch die Ausgabe des Kommandos

## 2.4 Umleitungen

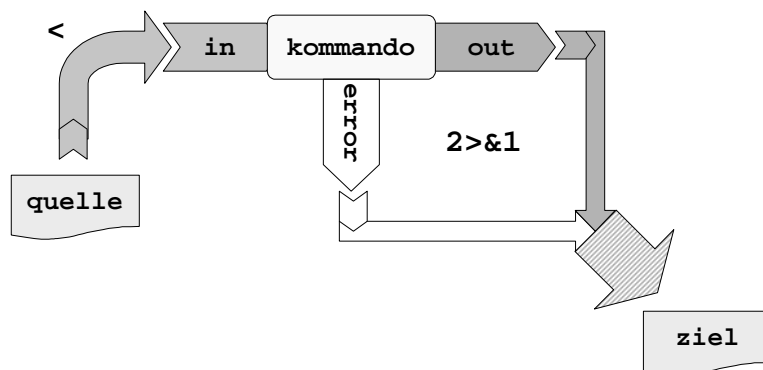
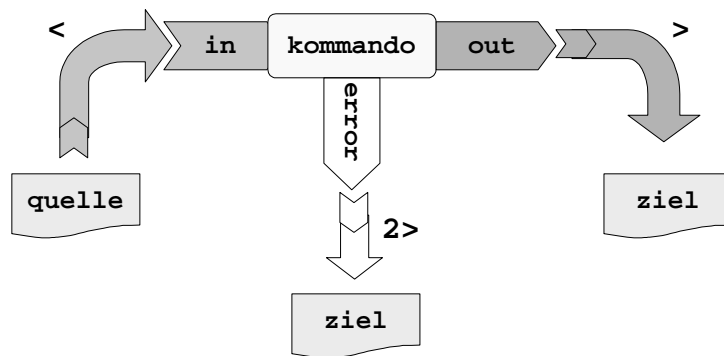
Ausgaben werden von der Shell nicht direkt auf den Bildschirm geschrieben, sondern in Kanäle. Diese Kanäle sind nicht fest, sondern können umgelenkt werden. Dadurch werden vielfältige flexible Kombinationen von Kommandos möglich.

Es gibt drei wichtige Standardkanäle:

- `stdout` Standardausgabekanal
- `stderr` Standardfehlerkanal
- `stdin` Standardeingabekanal



Was	Wie
stdin	kommando < quelle
stdout	kommando > ziel
stderr	kommando 2> ziel
stdin, stdout	kommando < quelle > ziel
stdout und stderr	kommando > ziel 2>&1
stdout, stderr	kommando > ziel <sub>out</sub> 2> ziel <sub>err</sub>



Sollen die bestehenden Ziele nicht überschrieben, sondern die Daten angehängt werden, so ist jeweils das >-Zeichen durch >> zu ersetzen.

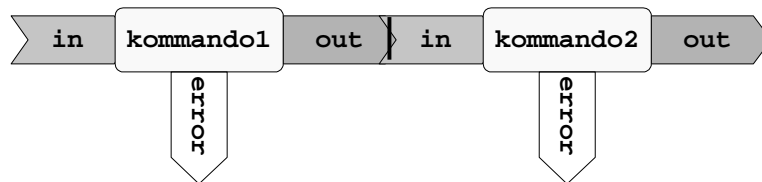
Beispiel: Das Kommando `echo` gibt den als Parameter übergebenen Text auf die Standardausgabe aus. In Kombination mit einer Umlenkung läßt sich so eine Datei anlegen: `echo > datei` legt eine leere Datei mit dem Namen `datei` im aktuellen Verzeichnis an.

## 2.5 Die Pipe

Das Pipe-Symbol `|` verknüpft die Standardausgabe eines Kommandos mit der Standardeingabe des nächsten Kommandos: Statt

- `kommando1 > datei`
- `kommando2 < datei`

einfach `kommando1 | kommando2`



## 2.6 Kombination von Kommandos

Jedes Kommando liefert einen Rückgabewert, der in der Variable  `$?`  abgelegt wird. Ausgeben läßt er sich etwa mit `echo $?`, wobei `0` für Erfolg (`TRUE`) steht und jeder von `0` verschiedene Wert für einen Fehler (`FALSE`). Der Befehl `true` ist immer erfolgreich, während `false` immer einen Fehler liefert.

Schreibweise	Bedeutung
<code>kommando<sub>1</sub> ; kommando<sub>2</sub></code>	Führt beide Kommandos nacheinander aus
<code>kommando<sub>1</sub>    kommando<sub>2</sub></code>	führt <code>kommando<sub>2</sub></code> nur aus, wenn <code>kommando<sub>1</sub></code> nicht erfolgreich war
<code>kommando<sub>1</sub> &amp;&amp; kommando<sub>2</sub></code>	führt <code>kommando<sub>2</sub></code> nur aus, wenn <code>kommando<sub>1</sub></code> erfolgreich war

## 3 Erste Kommandos

Neben den bereits kennengelernten Kommandos `ls` und `echo` sollen nun weitere, grundlegende Kommandos angegeben werden.

### 3.1 Das Hilfesystem

Zu vielen Themengebieten gibt es Hilfeseiten, sogenannte **Manual-Pages** oder kurz `man-Pages`. Sie lassen sich mittels

- `man [kapitel] schlagwort`

einsehen. Beendet werden kann mit `q`. Bei der Suche nach Schlagworten kann

- `apropos suchwort`

hilfreich sein.

### 3.2 Dateien lesen

- `cat`
  
- `less`
  
- `more`
  
- `tail`
  
- `head`

### 3.3 Informationen zu Dateien einholen

- `wc`
- `file`
- `ls`

## 4 Benutzerkommunikation

- `mail`
- `talk`
- `write`
- `mesg`

## 5 Benutzereigenschaften

- passwd
- id
- finger
- who
- groups

## 6 Das Dateisystem

### 6.1 Dateioperationen

- Datei anlegen: `echo > datei`
- Datei kopieren (copy): `cp datei neuedatei`
- Datei löschen (remove): `rm datei`
- Datei umbenennen (move): `mv datei neuername`
- Datei verschieben (move): `mv datei neuerort`

### 6.2 Verzeichnisoperationen

Verzeichnisse strukturieren die Dateien in ein hierarchisches System mit einer Wurzel und vielen Blättern. Über dieses hierarchische System sind i.W. alle Bestandteile eines Linux-Systems erreichbar, d.h. alle Laufwerke im weitesten Sinne, alle sonstigen Geräte, das Prozesssystem u.v.a.m. Für die Laufwerke stehen keine Laufwerksbuchstaben (C:\) zur Verfügung, alle Laufwerke sind über einen (im Prinzip beliebigen) Pfad innerhalb des Dateibaumes anzusprechen. Verzeichnistrennzeichen ist der **slash** /. (DOS/Windows: backslash \)

- in Verzeichnis wechseln (change directory): `cd verzeichnis`
  - eigener Verzeichniseintrag: `.`
  - Elternverzeichnis: `..`
  - Heimatverzeichnis des aktuellen Benutzers: `~`
  - Wurzelverzeichnis (root) `/`

- aktuelles Verzeichnis ausgeben lassen (print working directory): `pwd`
- Verzeichnis anlegen (make directory): `mkdir verzeichnis`
- Verzeichnis kopieren (copy): `cp -r verzeichnis neuesverzeichnis`
- Verzeichnis löschen (remove directory): `rmdir verzeichnis`
- Verzeichnis umbenennen (move): `mv verzeichnis neuename`
- Verzeichnis verschieben (move): `mv verzeichnis neuerort`

### 6.3 Verweise

Es gibt viele Gründe eine Datei unter mehreren Namen oder an verschiedenen Stellen ansprechen zu wollen. Dazu stellen die Dateisysteme das Konzept der Pseudonyme zur Verfügung. Es gilt prinzipiell zu unterscheiden:

- Symbolische Links (soft links)  
Sind über Dateisystemgrenzen hinweg und auf Verzeichnisse möglich, das Ziel der Verknüpfung ist mit `ls -l` sichtbar; es gibt keinen Zähler der angibt wieviele Stellen auf eine Datei verweisen
- Harte Links (hard links)  
nur innerhalb eines Dateisystems, keine harten Links auf Verzeichnisse, das Ziel ist nicht sichtbar; es gibt Zähler die angeben wieviele Stellen auf eine Datei verweisen

Angelegt werden Verweise beispielsweise mittels `ln -s datei neuename`

## 6.4 Zugriffsrechte

Jeder Benutzer eines Linux-Systems ist mindestens einer Gruppe zugeordnet — seiner **primären Gruppe**. Darüberhinaus kann jeder Benutzer prinzipiell in unbegrenzt vielen **weiteren Gruppen** Mitglied sein. Die Gruppenzugehörigkeit kann etwa durch `groups` erfragt werden; noch weitergehende Möglichkeiten bietet das Kommando `id`.

Jede Datei hat einen Besitzer (**owner**) und ist einer Gruppe (**group**) zugeordnet. Mit Hilfe dieser Zuordnung Datei-Besitzer-Gruppe werden die Zugriffsrechte verwaltet.

### 6.4.1 Normale Darstellung

Das Langformat eines Directory-Listings zeigt in der ersten Spalte die Zugriffsrechte:

`-rwxrwxrwx ...besitzer gruppe ...dateiname` Auf das erste Zeichen (`-` für reguläre Datei) folgen drei Gruppen zu je drei Zeichen mit folgender Bedeutung:

- erste Gruppe: Besitzer (**owner**)
- zweite Gruppe: zugeordnete Gruppe (**group**)
- dritte Gruppe: die anderen (**others**), die weder der Besitzer sind noch der Gruppe zugeordnet sind

`-rwxrwxrwx ... benutzer gruppe ... datei`

Benutzer	Gruppe	andere
u	g	o
a Alle		

#### Die Rechte im Einzelnen:

- Das Recht ist nicht gesetzt
- r Lesen
- w Schreiben
- x Datei: ausführen  
Verzeichnis: hinein wechseln

Zeichen	Bedeutung
r	lesen erlaubt
w	schreiben erlaubt
x	reguläre Datei: ausführen erlaubt Verzeichnis: wechseln in das Verzeichnis erlaubt
-	nicht gesetzt
u+s	<b>SUID-Bit:</b> führt Programm mit Rechten des Besitzers aus
g+s	<b>SGID-Bit:</b> bei Dateien: führt Programm mit Rechten der Gruppe aus <b>SGID-Bit:</b> bei Verzeichnissen: Jede unterhalb angelegte Datei wird der gleichen Gruppe zugeordnet wie das Verzeichnis selbst
t	<b>sticky-Bit:</b> Zum Löschen einer Datei wird das Schreibrecht auf der Datei selbst benötigt

### 6.4.2 Oktale Darstellung

Die Darstellung kann wesentlich vereinfacht werden, wenn den einzelnen Rechten unterschiedliche Wertigkeiten zugeordnet werden und diese dann für jede der drei Gruppen aufaddiert werden:

Zeichen	Wertigkeit	Zeichen	Wertigkeit
r	4	u+s	4
w	2	u+g	2
x	1	+t	1
-	0		

<code>-rwxrwxrwx</code>	<code>u+s</code>	<code>4</code>
<code>0421421421</code>	<code>g+s</code>	<code>2</code>
<code># + + +</code>	<code>+t</code>	<code>1</code>

Beispiel: Aus `-rwxrwxr-t` wird einfach der Zahlencode 1775

### 6.4.3 Zugriffsrechte ändern

- Zugriffsrechte (change mode): `chmod MODE datei`
- Ändern der Gruppenzuordnung (change group): `chgrp`
- Ändern des Besitzers (change owner): `chown` (darf nur root)

## 6.5 Gerätedateien

Alle Geräte sind über das Dateisystem ansprechbar. Geräte sind unterhalb `/dev/` eingeordnet und systematisch benannt. Diskettenlaufwerke heißen `fd0` (floppydrive) und `fd1`, Festplatten entsprechend `hd1` (harddisk) usf.

Das erste Zeichen eines Directory-Listings im Langformat zeigt an, um welche Dateiart es sich genau handelt:

Zeichen	Art der Datei
-	reguläre Datei
d	Verzeichnis
l	Verknüpfung
c	zeichenorientiertes Gerät
b	blockorientiertes Gerät

Neben den erwähnten „realen“ Geräten gibt es eine ganze Reihe Geräte mit besonderer Funktion:

- `/dev/null` ist ein Daten-Nirwana: Alle hierhin gesendeten Dateien sind für immer verloren
- `/dev/zero` ist eine unerschöpfliche Datenquelle, aus der beständig 0-Bytes gelesen werden können

## 6.6 Dateien suchen und finden

- `locate` sucht in einer Datenbank, schnell aber nicht immer aktuell
- `find` bietet **sehr viele Suchmöglichkeiten**

## 7 Editoren

### 7.1 Der Editor vi

Der Editor *vi* ist ein Standardeditor, der zwar nicht unbedingt einfach zu bedienen ist, aber eine ganze Reihe von Vorzügen aufweist, die ihn zu einem Werkzeug machen dessen Grundzüge unbedingt bekannt sein sollten. Eine ganze Reihe von Programmen (`crontab`, ...) ruft den `vi` direkt auf, bei einigen Spezialaufgaben (Erstellen von Access-Control-Listen) ist er unverzichtbar.

Der Editor kennt zwei verschiedene Modi:

- Befehlsmodus (Startmodus)
- Eingabemodus

Vom Eingabemodus aus führt Drücken von `Esc` in den Befehlsmodus, vom Befehlsmodus in den Eingabemodus beispielsweise `i` (für `insert`).

#### 7.1.1 Editierbefehle

Befehl	Wirkung
<code>i</code>	Text vor Cursor einfügen, Wechsel in Eingabemodus
<code>x</code>	Zeichen unter dem Cursor löschen
<code>dnr</code>	Anzahl <i>nr</i> Zeichen löschen
<code>dw</code>	Wort löschen
<code>dd</code>	Zeile löschen

#### 7.1.2 Suchbefehle

Befehl	Wirkung
<code>/muster</code>	Vorwärts nach <i>muster</i> suchen
<code>?muster</code>	Rückwärts nach <i>muster</i> suchen
<code>/</code>	letzte Suche vorwärts wiederholen
<code>?</code>	letzte Suche rückwärts wiederholen

#### 7.1.3 Befehle zum Beenden und Speichern

Befehl	Wirkung
<code>:w</code>	Datei speichern
<code>:x</code>	Datei speichern und beenden
<code>:q</code>	Datei nicht speichern und beenden
<code>!</code>	in Kombination mit den anderen Befehlen: Befehl erzwingen

### 7.2 mcedit

- `mcedit`

## 8 Werkzeuge und andere Befehle

- (get regular expression) `grep`
  
- `cut`
  
- `sort`
  
- `uniq`
  
- `date`
  
- (disk free): `df`
  
- (disk usage): `du`

