

Betriebssysteme

WS 2015/16

Hans-Georg Eßer

Foliensatz A:

- Einführung
- Installation der Linux-VM
- Shell-Grundlagen, Dateiverwaltung

v1.2, 2015/09/02

20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-1

Zur Veranstaltung (1)

Veranstaltung kombiniert:

Theorie und **Praxis** der Betriebssysteme

Service / Web-Seite: <http://fom.hgesser.de>

- Folien und Praktikumsaufgaben
- Vorlesungs-Videos („test, test“)
- Probeklausur gegen Semesterende
- Folien auch im Online-Campus

20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-3

Über den Dozenten

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)
Dipl.-Inform. (RWTH Aachen, 2005)
Fachjournalist (FJS Berlin, 2006)
Promotion (Univ. Erlangen-Nürnberg, 2015)
- Chefredakteur Linux-Zeitschrift (seit 2000)
und Autor diverser Computerbücher
- LPI-zertifiziert (LPIC-1 und LPIC-2)
- seit 2006 Dozent (u. a. FOM, HS München, TH Nürnberg,
Univ. Erlangen-Nürnberg): Betriebssysteme, Rechner-
architektur, IT-Infrastruktur, Informatik-Grundlagen, System-
programmierung, Betriebssystem-Entwicklung, IT-Sicherheit



20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-2

Zur Veranstaltung (2)

Hilfreiche Vorkenntnisse:

- **Linux-Shell** – Benutzung der Standard-Shell *bash* unter Linux
→ Bash-Crashkurs
- **C** – Grundlagen der Programmierung in C
(oder C++, C#, Java)
- **Rechnerarchitekturen** – grober Aufbau eines
Computers (Prozessor, Hauptspeicher,
Peripherie etc.)

20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-4

Praktikum:

- Systemadministration unter Linux
- Praktische Programmier-Beispiele zum Theorieteil in **C** umsetzen

Prüfung und Benotung:

1. Lernfortschrittskontrolle (LFK)
2. Klausur über 120 Minuten

Fragen:

- direkt in der Vorlesung (Handzeichen)
- oder danach oder per E-Mail

Einführung und Motivation

Linux-Administration

- Nutzen von Shell-Befehlen
- Standard-Datei- und -Verzeichnis-Operationen
- Editor vi
- Shell-Variablen, Unix-Filter-Programme
- Jobs und Prozesse
- Software-Verwaltung
- Einrichtung von Festplatten (-Partitionen)
- Dateisuche, Auskunft
- Benutzer- und Gruppen-Rechte

LPI-Zertifizierung (1)**LPI Certified Junior Level Linux Professional**

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, backup & restore, shutdown & reboot
- Install and configure a workstation (including X) and connect it to a LAN, or a stand-alone PC via modem to the Internet.

**LPI Certified Advanced Level Linux Professional**

- Administer a small to medium-sized site
- Plan, implement, maintain, keep consistent, secure, and troubleshoot a small mixed (MS, Linux) network, including a LAN server (samba), Internet Gateway (firewall, proxy, mail, news), Internet Server (webserver, FTP server)
- Supervise assistants
- Advise management on automation and purchases

**LPI Certified Senior Level Linux Professional**

- hauptsächlich: LDAP
- dazu: verschiedene Spezialisierungen
 - LPI 302: Mixed Environments
 - LPI 303: Security
 - LPI 304: Virtualization and High Availability
- es kommen vielleicht noch weitere

Im Rahmen dieser
Vorlesung: teilweise (!)
Vorbereitung auf die
LPIC-1-Prüfungen
101 und 102



- keine grafischen Werkzeuge – auch wenn es welche gibt
- also: nicht YaST & Co., sondern Kommandozeilentools, Konfigurationsdateien, Shell-Skripte
- verstehen, was im Hintergrund abläuft

- Für die Zertifizierung: Stoff aus dieser Veranstaltung reicht nicht
- insbesondere: „passives Konsumieren“ reicht nicht
- Prüfungen sind so gestaltet, dass Administratoren sie leicht bestehen können
→ üben, üben, üben :)
- Kosten: 145 € (+ USt) pro Prüfung (LPIC-1 = 2 Prüfungen)

- Windows setzt auch bei der Administration überwiegend auf grafische Tools
- „GUI-Administration“ leichter (schneller) zu erlernen, bietet aber weniger Möglichkeiten, wenn etwas schief geht
- Windows ist im Rahmen dieser Vorlesung kein Thema

„Klicken Sie auf Schließen.“

- Im Theorieteil nicht: „Wie bediene ich ... ?“, sondern:
„Wie und warum funktioniert ... intern?“
 - Konsequenzen für Anwendungsentwickler
 - Sicherheitsprobleme
 - Auswahl eines geeigneten Betriebssystems
- ... und das Thema ist auch an sich spannend

- Schützt Hardware vor direkten Zugriffen
(→ defekte oder bösartige Software)
- Befreit Software vom Zwang, die Hardware im Detail zu kennen
- Zulassen mehrerer Benutzer und Abgrenzung (Multi-user)
- Parallelbetrieb mehrerer Anwendungen (Multi-tasking): faire Aufteilung der Ressourcen

- Abstraktionsschicht zwischen Hardware und Programmen (→ virtuelle Maschine)
- Verwaltung der vorhandenen Ressourcen
- Einheitlicher Zugriff auf Geräte einer groben Kategorie, z. B.:
 - *Datenträger* (Plattenpartition, CD, DVD, Diskette, USB-Stick, Netzwerk-Volume)
 - *Drucker* (PostScript-Laser, Etikettendrucker, Billig-Tintenstrahler, ...)

- Virtualisierung des Speichers
 - Anwendungen müssen nicht wissen, wo sie im Hauptspeicher liegen
 - Speicher über phys. RAM hinaus verfügbar (Swap etc.)

Desktop-PC – die Standardaufgabe, Intel & Co.

- Anwendungsprogramme (Office, Grafik, kaufmännische Software etc.)
- Internet-Zugang und Web-basierte Anwendungen (WWW, E-Mail, File Sharing, ...)
- Datenbank-Client
- Software-Entwicklung
- Multimedia



Server-PC

Häufig ähnliche Hardware wie Desktop-PC, aber ganz andere Einsatzgebiete:

- Web- / FTP- / Mail-Server (Internet oder Intranet)
- Datenbank-Server
- „Number Crunching“ bzw. High Performance Computing (oft: Cluster)

Industrieanwendungen

- Robotersteuerung
- automatische Navigation
- Temperaturregelung
- Motorenkontrolle
- Herzschrittmacher

→ **Echtzeit-Betriebssysteme**
(real time operating systems)



Bild: Wikipedia, KUKA Schweißanlagen

Embedded systems (ohne Echtzeit-Ansprüche)

- Mobiltelefone, PDAs, mobile MP3/Video-Player
- Fernseher, Videorekorder, DVD-Player
- DSL-WLAN-Router (mit Firewall etc.)
- Taschenrechner
- Videospiel-Konsolen
- Geldautomaten

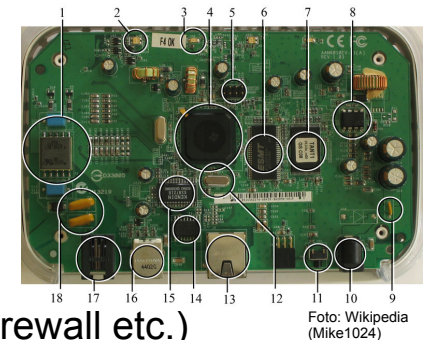


Foto: Wikipedia (Mike1024)

Beim Programmieren tauchen häufig Probleme in zwei Bereichen auf:

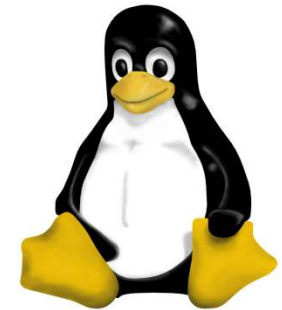
- **Zuverlässigkeit**

Software tut nicht das, was sie soll;
unerwartetes Verhalten;
mangelnde Fehlertoleranz

- **Sicherheit**

Software ist nicht geschützt vor Angriffen
durch Dritte

- Etabliertes Standardsystem für sehr viele Plattformen (PC Desktop / Server, Embedded etc.)
- vor allem auf Servern weit verbreitet
- Offene Kernel-Quellen:
 - nachlesen, wie etwas geht
 - ändern, was nicht gefällt
- Image eines virtuellen Linux-PCs für VMware / VirtualBox



- Funktionsweise des Betriebssystems nicht klar
→ fehlerhaft programmierte Anwendungen, z. B.
- Race Conditions
 - Buffer Overflows

Darum verstehen und lernen, wie
Betriebssysteme intern arbeiten

**Grundlagenbuch Linux**

Grundlagen, Techniken, Lösungen
(Eßer, Dölle)

Data Becker, 2007

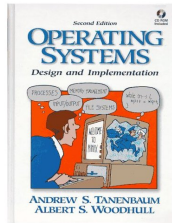
→ als PDF-Dokument im Campus-System

**Linux 2011**

Debian, Fedora, openSUSE, Ubuntu
(Kofler)

Addison-Wesley, 2010

49,80 €; E-Book: 39,80 €



Operating Systems
Design and Implementation
(Tanenbaum, Woodhull)
Prentice Hall
(englisch)



Betriebssysteme
Ein Lehrbuch mit Übungen zur System-
programmierung in Unix/Linux (Ehses et al.)
ISBN 3-8273-7156-2
Pearson Studium, 30 Euro

20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

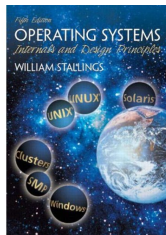
Folie A-25

Gliederung

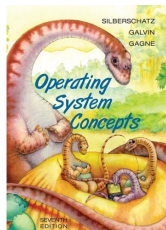
20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-27



Operating Systems
Internals and Design Principles
(Stallings)
Prentice Hall, ca. 80 Euro
(englisch)



Operating System Concepts
(Silberschatz, Galvin, Gagne)
Wiley, ca. 52 Euro
(englisch)

20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-26

BS Praxis

- **Arbeiten mit der Shell**
- Dateiverwaltung
- Filter
- Prozesse und Jobs
- Software-Verwaltung
- Partitionen, formatieren, mounten
- Auskunft
- User, Gruppen, Zugriffsrechte

BS Theorie

- Prozesse und Threads
- Polling und Interrupts
- System Calls
- Systemprogrammierung
- Scheduling
- Synchronisation
- Speicherverwaltung

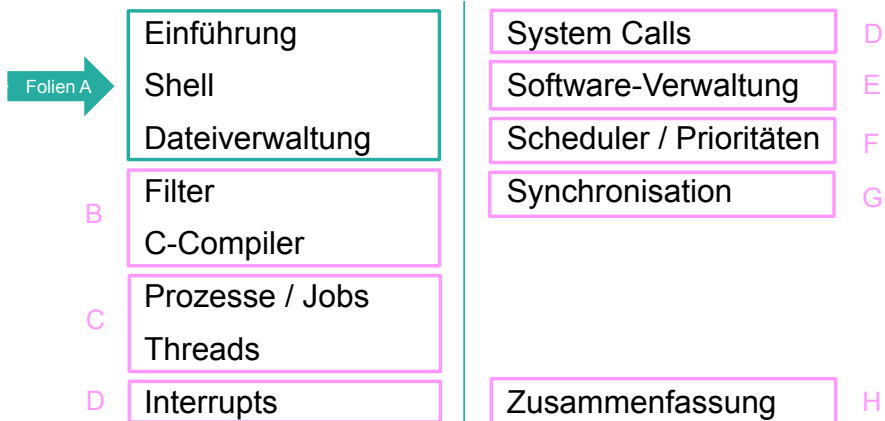
20.08.2015

Betriebssysteme, WS 2015/16, Hans-Georg Eßer

Folie A-28

	Shell	Dateiverwaltung	Filter	C-Compiler	Prozesse / Jobs	Threads	Interrupts	System-Calls	Scheduler / Priorit.	Synchronisation	Speicherverw.	Software-Verw.	Partitionen	Auskunft	User, Gr., Rechte
BS Theorie					X	X	X	X	X	X	X				
Systemprog.				X	X	X		X	X						
Linux-Progr.				X	X	X		X	X						
BS Praxis	X	X	X	X	X			X	X			X	X	X	X

Debian Linux in der virtuellen Maschine



Praxisteil

- Installation VirtualBox auf Ihrem Notebook / Netbook
- Import einer virtuellen Maschine mit Debian Linux 6
- erste Experimente mit dem installierten Linux

Praxisteil (Demonstration)

- Arbeiten mit der Shell
- Verzeichnisnavigation, -Listings
- Dateien kopieren, umbenennen, verschieben
- Verzeichnisse erstellen, löschen etc.
- Dateien öffnen
- Der Editor „vi“

- Vor dem \$, >, # meist Hinweise auf Benutzer, Rechner, Arbeitsverzeichnis

```
[esser@macbookpro:BS] $
```

```
root@quad:~#
```

- esser, root: Benutzername; individuell
- macbookpro, quad: Rechnername
- BS, ~: Arbeitsverzeichnis, je nach Prompt-Einstellung auch in voller Länge (z. B. /home/esser/Daten/FOM/WS2014/BS)
- ~ = „Home-Verzeichnis“ des Benutzers

- Shell zeigt durch **Prompt** an, dass sie bereit ist, einen Befehl entgegen zu nehmen
- Prompts können verschieden aussehen:
 - ... \$ _ : Anwender-Prompt, nicht-privilegiert
 - ... > _ : Anwender-Prompt, nicht-privilegiert
 - ... # _ : Root-Prompt, für den Administrator

- Am Prompt Befehl eingeben und mit [Eingabe] abschicken
- Shell versucht, (in der Regel) erstes Wort als Kommandoname zu interpretieren:
 - Alias? (→ nicht in dieser Vorlesung)
 - Shell-interne Funktion? (→ nicht in dieser Vorles.)
 - eingebautes Shell-Kommando? (z. B. cd)
 - externes Programm? (Suche in Pfad)

- Beispiel: Aktuelles **Arbeitsverzeichnis** anzeigen (`pwd` = **p**rint **w**orking **d**irectory)

```
[esser@quad:~]$ pwd
/home/esser
[esser@quad:~]$ _
```

- Nach Abarbeiten des Befehls (oft: mit einer „Antwort“) erscheint wieder der Prompt – Shell ist bereit für nächstes Kommando

- **Inhaltsverzeichnis** anzeigen: `ls` (**l**ist)
- bezieht sich immer auf das aktuelle Arbeitsverzeichnis (Alternative: Ort als Parameter angeben)

```
[esser@quad:~]$ ls
bahn-2012-11-06.pdf      bh-win-04-kret.pdf
buch_kap08.pdf          bv-anleitung.pdf
bz2.pdf
[esser@quad:~]$ ls /tmp
cvcd  kde-esser  ksocket-esser  orbit-esser
ssh-vrUNLb1418  virt_1111
[esser@quad:~]$ _
```

- Mehrere Befehle auf einmal abschicken: mit Semikolon ; voneinander trennen

```
[esser@quad:~]$ pwd; pwd
/home/esser
/home/esser
[esser@quad:~]$ _
```

- Inhalt mit mehr Informationen: `ls -l`

```
[esser@quad:~]$ ls -l
-rw-r--r-- 1 esser users 29525 Nov 09 14:11 bahn-2012-11-09.pdf
-rw-r--r-- 1 esser users 74520 Apr 10 2004 bh-win-04-kret.pdf
-rw-r--r-- 1 esser users 856657 Oct 21 2005 buch_kap08.pdf
-rw-r--r-- 1 esser esser 738570 Mar 17 20:29 bv-anleitung.pdf
-rw-r--r-- 1 esser users 123032 Sep 22 2003 bz2.pdf
[esser@quad:~]$ _
```

- Ausgabe enthält zusätzlich:
 - Zugriffsrechte (`-rw-r--r--` etc.) → später
 - Dateibesitzer und Gruppe (`esser, users`) → später
 - Größe und Datum/Zeit der letzten Änderung

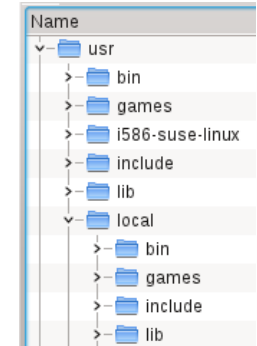
- Leere Datei erzeugen (für Experimente): `touch`

```
[esser@quad:~]$ touch Testdatei
[esser@quad:~]$ ls -l Testdatei
-rw-r--r-- 1 esser esser 0 Apr  7 13:58 Testdatei
[esser@quad:~]$ _
```

- Datei hat Größe 0

Grundlagen (1)

- Linux kennt keine „Laufwerksbuchstaben“ (C:, D: etc.)
- Wurzelverzeichnis heißt /
- Pfadtrenner: auch / – d. h.:
/usr/local/bin ist das Verzeichnis bin im Verzeichnis local im Verzeichnis usr.
(wie bei Webadressen)



- Fehlermeldungen: Unbekanntes Kommando

```
[esser@quad:~]$ fom
No command 'fom' found, did you mean:
Command 'fim' from package 'fim' (universe)
Command 'gom' from package 'gom' (universe)
Command 'fop' from package 'fop' (universe)
Command 'fdm' from package 'fdm' (universe)
Command 'fpm' from package 'fpm2' (universe)
[...]
fom: command not found
[esser@quad:~]$ _
```

- Meldung kann auch deutschsprachig sein

Grundlagen (2)

- Weitere Datenträger erscheinen in Unterordnern
 - Beispiel: DVD mit Dateien zum Kurs hat Volume-Name BS-ESSER
 - Datei `test.txt` auf oberster DVD-Verzeichnisebene ist als `/media/BS-ESSER/test.txt` erreichbar (Windows: `e:\test.txt`)
 - Datei `Software/index.html` der DVD entsprechend als `/media/BS-ESSER/Software/index.html` (Windows: `e:\Software\index.html`)

Grundlagen (3)

- Für private Nutzerdaten hat jeder Anwender ein eigenes **Home-Verzeichnis**, das i. d. R. unterhalb von `/home` liegt, z. B. `/home/esser`.
- Die Tilde `~` ist immer eine Abkürzung für das Home-Verzeichnis
 - funktioniert auch in zusammengesetzten Pfaden
 - `~/Daten/brief.txt` statt `/home/esser/Daten/brief.txt`

Grundlagen (5)

- Zwei Spezialverzeichnisse in jedem Ordner
 - `..` ist das Verzeichnis eine Ebene höher (von `/usr/local/bin` aus ist `..` also `/usr/local`)
 - `.` ist das aktuelle Verzeichnis
- Pfade kann man **absolut** und **relativ** zusammen bauen
 - absoluter Pfad beginnt mit `/`
 - relativer Pfad nicht; er gilt immer ab dem aktuellen Arbeitsverzeichnis

Grundlagen (4)

- Ausnahme: Das Home-Verzeichnis des Systemadministrators `root` ist nicht `/home/root`, sondern `/root`
- Der Trick mit der Tilde `~` funktioniert aber auch für `root`
- Warum? `/home` könnte auf einer separaten Partition liegen und bei einem Fehlstart nicht verfügbar sein

Verzeichnisnavigation

- Kommando `cd` (change directory) wechselt in ein anderes Verzeichnis
- Zielverzeichnis als Argument von `cd` angeben – wahlweis mit relativem oder absolutem Pfad

```
[esser@quad:~]$ pwd
/home/esser
[esser@quad:~]$ cd /home ; pwd
/home
[esser@quad:home]$ cd .. ; pwd
/
[esser@quad:/]$ _
```

Datei kopieren

- Kommando `cp` (**copy**) kopiert eine Datei
- Reihenfolge: `cp Original Kopie`

```
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[esser@quad:tmp]$ cp test.dat kopie.dat
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  8 12:17 kopie.dat
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[esser@quad:tmp]$ _
```

! Kopie erhält aktuelles Datum/Zeit

Datei verschieben

- Kommando `mv` (**move**) verschiebt eine Datei
- Reihenfolge: `mv AltName NeuerOrdner/`

```
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[esser@quad:tmp]$ mv test.dat /home/esser/
[esser@quad:tmp]$ ls -l
[esser@quad:tmp]$ ls -l /home/esser/
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[...]
```

! Verschieben ändert Datum/Zeit nicht

Datei umbenennen

- Kommando `mv` (**move**) benennt eine Datei um
- Reihenfolge: `mv AltName NeuName`

```
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[esser@quad:tmp]$ mv test.dat neu.dat
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 neu.dat
[esser@quad:tmp]$ _
```

! Umbenennen ändert Datum/Zeit nicht

Datei löschen

- Kommando `rm` (**remove**) löscht eine Datei

```
[esser@quad:tmp]$ ls -l
-rw-r--r-- 1 esser wheel 1501 Apr  5 11:37 test.dat
[esser@quad:tmp]$ rm test.dat
[esser@quad:tmp]$ ls -l
[esser@quad:tmp]$ _
```

Mehrere Dateien

- Einige Befehle akzeptieren mehrere Argumente, z. B.

- mv (beim Verschieben in anderen Ordner)
- rm

- Beispiele:

```
[esser@quad:tmp]$ mv datei1.txt datei2.txt Ordner/  
[esser@quad:tmp]$ rm datei3.txt datei4.txt datei5.txt  
[esser@quad:tmp]$ _
```

- Löschbefehl mit Wildcards zu gewagt?
→ vorher mit echo testen:

```
[esser@quad:Downloads]$ echo rm *.zip  
rm Logo_a5_tif.zip Uebung1.zip c32dwenu.zip  
ct.90.01.200-209.zip ct.90.12.130-141.zip  
ct.91.02.285-293.zip ct.91.12.024-025-1.zip  
ct.91.12.024-025.zip ct.92.08.052-061.zip  
ix.94.03.010-011.zip ix.94.07.068-071.zip  
[esser@quad:Downloads]$ rm *.zip  
[esser@quad:Downloads]$ _
```

Wildcards (*, ?)

- Bei Befehlen, die mehrere Argumente akzeptieren, können Sie auch Wildcards verwenden:
 - * steht für beliebig viele (auch 0) beliebige Zeichen
 - ? steht für genau ein beliebiges Zeichen

- Beispiele:

```
[esser@quad:~]$ ls -l ??????.pdf  
-rw-r--r-- 1 esser staff 79737 Apr 2 01:18 RegA4.pdf  
-rw-r--r-- 1 esser staff 132246 Apr 4 18:02 paper.pdf  
[esser@quad:~]$ rm /tmp/*  
[esser@quad:~]$ _
```

- Das letzte Beispiel verrät etwas über das Auflösen der Wildcards
 - Wenn Sie `rm *.zip` eingeben, startet die Shell *nicht* `rm` mit dem Argument „*.zip“
 - Die Shell sucht im aktuellen Verzeichnis alle passenden Dateien und macht jeden Dateinamen zu einem Argument für den `rm`-Aufruf.
 - Es wird also
`rm Logo_a5_tif.zip Uebung1.zip
c32dwenu.zip ct.90.01.200-209.zip ...
aufgerufen.`

Mit Verzeichnissen können Sie ähnliche Dinge tun wie mit Dateien

- Verzeichnis erstellen
- (leeres!) Verzeichnis löschen
- Verzeichnis umbenennen oder verschieben
- Verzeichnis rekursiv (mit allen enthaltenen Dateien und Unterordnern) löschen

Verzeichnis erstellen

- Kommando `mkdir` (**make directory**) erzeugt ein neues (leeres) Unterverzeichnis

```
[esser@quad:tmp]$ ls -l
[esser@quad:tmp]$ mkdir unter
[esser@quad:tmp]$ ls -l
drwxr-xr-x  2 esser  wheel   68 Apr  8 14:28 unter
[esser@quad:tmp]$ cd unter
[esser@quad:unter]$ ls -l
[esser@quad:unter]$ cd ..
[esser@quad:tmp]$ _
```

! Kurzform `md` für `mkdir` nicht immer vorhanden → vermeiden

Verzeichnis löschen

- Kommando `rmdir` (**remove directory**) löscht ein leeres (!) Unterverzeichnis

```
[esser@quad:tmp]$ touch unter/datei
[esser@quad:tmp]$ rmdir unter
rmdir: unter: Verzeichnis nicht leer
[esser@quad:tmp]$ rm unter/datei
[esser@quad:tmp]$ rmdir unter
[esser@quad:tmp]$ _
```

! Kurzform `rd` für `rmdir` nicht immer vorhanden → vermeiden

Verzeichnis umbenennen / verschieben

- funktioniert wie das Umbenennen / Verschieben von Dateien
- gleicher Befehl: `mv`, wieder zwei Varianten:
 - `mv Verzeichnis NeuerName`
 - `mv Verzeichnis AndererOrdner/`

Verzeichnis rekursiv löschen

- Kommando `rm` (**r**emove) hat eine Option `-r` zum rekursiven Löschen:

```
[esser@quad:tmp]$ mkdir a; mkdir a/b; mkdir a/b/c
[esser@quad:tmp]$ touch a/b/c/datei
[esser@quad:tmp]$ rmdir a
rmdir: a: Verzeichnis nicht leer
[esser@quad:tmp]$ rm -r a
[esser@quad:tmp]$ _
```

! Vorsicht beim rekursiven Löschen: „Was weg ist, ist weg“

- Undelete = Löschen rückgängig machen
 - gibt es unter Linux nicht
 - Wiederherstellung von gelöschten Dateien mit Profi-Tools möglich, wenn Computer nach dem Löschen sofort ausgeschaltet wurde
 - solche Tools stellen aber sehr viele Dateien wieder her → enormer Aufwand, anschließend die gesuchte Datei zu finden; u. a. sind die Dateinamen dauerhaft verloren

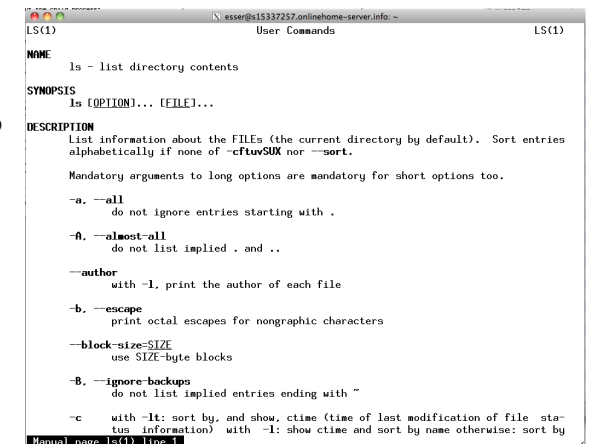
→ vor `rm -r ...` mehrfach prüfen ...

- **Argumente:** z. B. Dateinamen; beziehen sich oft auf Objekte, die manipuliert werden sollen
- **Optionen:** verändern das Verhalten eines Befehls
 - bei den meisten Befehlen zwei Varianten:
 - kurze Optionen: `-a`, `-b`, `-c`, ...
→ lassen sich kombinieren: `-abc` = `-a -b -c`
 - lange Optionen: `--ignore`, `--force`, `--all` etc.
 - Beispiel: `-r` bei `rm`

- Zu den meisten Kommandos gibt es eine sog. Manpage, die Sie über

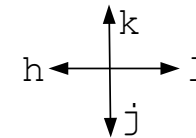
`man kommando`
abrufen

- Beispiel:
`man ls`



- Standard-Editor auf allen Unix-Systemen (und damit auch Linux): **vi** (visual editor)
- gewöhnungsbedürftige Bedienung
- zwei Betriebsarten
 - **Befehlsmodus** (nach Start aktiviert; Normalmodus)
 - **Bearbeitungsmodus**
- **vi** aus Versehen gestartet? Verlassen ohne Speichern von Änderungen mit **[Esc] :q!**

- Wechseln in den Bearbeitungsmodus: **i**, **I**, **a**, **A**
 - **i**: Text vor dem Cursor einfügen
 - **a**: Text nach dem Cursor einfügen
 - **I**: Text am Zeilenanfang einfügen
 - **A**: Text am Zeilenende einfügen
- Bearbeitungsmodus verlassen: **[Esc]**
- Navigieren im Text:
Cursortasten oder:



- Warum Umgang mit **vi** lernen?
 - auf jedem – noch so minimalistischen – Unix-System ist ein **vi** installiert (kleines Programm):


```
[esser@quad:~]$ ls -l /usr/bin/vi /usr/bin/emacs
-rwxr-xr-x 1 root root 5502096 Nov  9  2008 /usr/bin/emacs
-rwxr-xr-x 1 root root  630340 Oct 17  2008 /usr/bin/vi
```
 - läuft im Terminal → hilfreich bei Remote-Zugriff
 - Bei Problemen (Plattenfehler, nicht alle Dateisysteme verfügbar) sind andere Editoren evtl. nicht erreichbar, **vi** vielleicht doch → gilt leider nicht mehr für aktuelle Linux-Versionen
 - Thema ist LPI-prüfungsrelevant

- Zeichen / Text löschen:
 - im Bearbeitungsmodus mit **[Rückschritt]** und **[Entf]**, wie aus anderen Editoren bekannt
 - im Befehlsmodus mehrere Möglichkeiten:
 - **x** löscht Zeichen unter Cursor
 - **X** löscht Zeichen links von Cursor
 - **dw** löscht ab Cursor-Position bis Anfang des nächsten Worts
 - **dd** löscht aktuelle Zeile
 - vorab Zahl: Mehrfachausführung (**15dd**: 15 Zeilen)

- Speichern und beenden
 - Immer zuerst in den Befehlsmodus
→ im Zweifelsfall einmal [Esc] drücken
 - Speichern: :w
 - Speichern (erzwingen): :w!
 - Beenden (klappt nur, wenn Text seit letztem Speichern nicht verändert wurde): :q
 - Beenden erzwingen (ohne speichern): :q!
 - Speichern und beenden: :wq (oder: ZZ ohne „!“)

- Rückgängig machen / wiederherstellen
 - Letzte Änderung rückgängig machen: u (undo)
 - geht auch mehrfach: u, u, u, ...
 - ... und mit Mehrfachausführung: 3u macht die letzten drei Änderungen rückgängig
 - Einen Undo-Schritt aufheben: [Strg]+r :redo
 - mehrfaches Redo: z. B. 3 [Strg]+r

- Suche im Text
 - Vorwärtssuche: / und Suchbegriff, dann [Eingabe]
 - Sprung zum nächsten Treffer: n (next)
 - Rückwärtssuche: ? und Suchbegriff, dann [Eingabe]
 - Sprung zum nächsten Treffer: n
 - Wechsel zwischen Vorwärts- und Rückwärtssuche:
einfach / bzw. ? , dann Eingabe und mit n weiter
(in neuer Richtung) suchen

- Copy & Paste: Kopieren ...
 - yw (ab Cursorposition bis Wortende)
 - y\$ (ab Cursorposition bis Zeilenende)
 - yy (ganze Zeile)
 - 3yy (drei Zeilen ab der aktuellen)
- ... und Einfügen
 - P (fügt Inhalt des Puffers an Cursorposition ein)
- Cut & Paste
 - Löschen mit dd, dw etc.; dann einfügen mit P

- Copy & Paste mit der Maus
 - Wenn Sie die grafische Oberfläche verwenden, geht es auch mit der Maus:
 - Kopieren: Mauszeiger auf 1. Zeichen, klicken (und gedrückt halten), zum letzten Zeichen ziehen, loslassen
 - Einfügen: Cursor zu Ziel bewegen, dann (im Einfügemodus!) die mittlere Maustaste drücken
 - Bei beiden Schritten muss man je nach `vi`-Version evtl. die [Umschalt]-Taste drücken

- Aufgabenblatt
 - Umgang mit Dateien und Verzeichnissen
 - Editor vi

- Datei im Editor öffnen:
`[esser@quad:~] vi Dateiname`
- zweite Datei an Cursorposition hinzuladen:
`:read Dateiname`
(im Befehlsmodus!)