Einführung in die Kommandozeile

Stefan W. Kleyer

Ubucon Heidelberg

12.10.2013

Zitat (*IRC*)»<ma drarua> OH MAN!!!!.. Gerade läuft eine Kollegin bei mir vorbei und sieht das offene irc Fenster. Guckt mich an und sagt:"Bhoa du kannst programmieren .respekt"....und die ist eine ganze Gehaltsstufe über mir«

Was könnt ihr?

Was wollt ihr?

- Grundlagen
- Installieren von Paketen
- Dateiverwaltung
- `mount` und statisches Einbinden
- Umleitungen
- Informationen über das System
- Shell-Programmierung

Fahrplan

1. Pflichtteil
   1. Grundlagen
   2. Hilfen
   3. Weitere Befehle
   4. `sudo`
   5. Terminal-Editor `nano`
2. Zweiter Teil
   1. Installieren von Paketen
   2. Informationen über das laufende System
   3. Umleitungen
   4. Entpacken
3. Additum

- Dateiverwaltung
- Netzwerk
- Systemüberwachung
- Downloaden und Webbrowser
- `mount` und statisches Einbinden
- Shell-Programmierung

Programme für bash

Gnome	KDE	Xfce	LXDE
gnome-terminal	Konsole	Xfce-Terminal	LXTerminal

Alle diese Programme benutzen *bash*, es ist also prinzipiell irrelevant, welches man benutzt.

Gibt es ein Limit an offenen Terminals?
$\rightarrow$ Nein, im Allgemeinen nicht.

Grundlagen

Im Folgenden werden erklärt:

- ls
- cd
- clear
- pwd
- file

Der Anfang

Startet man ein Terminal, sieht das in etwa so aus:

```
1  benutzer@pcname:~$
```

"`ls`" - *list*:

```
1  benutzer@pcname:~$ ls
2  Desktop         Documents       Downloads
3  Music           Pictures        Public
```

Homeverzeichnis!?

"cd" - *change directory*:

```
1  benutzer@pcname:~$ cd Documents
2  benutzer@pcname:~/Documents$
3  benutzer@pcname:~$ ls
4  Beispiel.txt   Beispiel2.txt
```

Ebene nach oben?

```
benutzer@pcname:~/Documents$ cd ..
benutzer@pcname:~$ ls
Desktop         Documents       Downloads
Music           Pictures        Public
```

Von irgendwo ins Homeverzeichnis?

Nehme

```
benutzer@pcname:~/Documents$ cd ~
benutzer@pcname:~$ ls
Desktop         Documents       Downloads
Music           Pictures        Public
```

oder

```
benutzer@pcname:~/Documents$ cd --
benutzer@pcname:~$ ls
Desktop         Documents       Downloads
Music           Pictures        Public
```

Nun sieht unser Terminalfenster bereits so aus:

```
benutzer@pcname:~$ ls
Desktop         Documents       Downloads
Music           Pictures        Public
benutzer@pcname:~$ cd Documents
benutzer@pcname:~/Documents$
benutzer@pcname:~$ ls
Beispiel.txt   Beispiel2.txt
benutzer@pcname:~$ cd ..
benutzer@pcname:~$ ls
Desktop         Documents       Downloads
Music           Pictures        Public
```

Nehme

```
benutzer@pcname:~$ clear
```

...

clear

. . . dann sieht das in etwa so aus:

```
1  benutzer@pcname:~$
12
```

Print Working Directory

Rückschau:

```
1  benutzer@pcname:~$ ls
2  Desktop         Documents       Downloads
3  Music           Pictures        Public
```

⇒ Homeverzeichnis!?

“print working directory” - pwd:

```
1  benutzer@pcname:~$ pwd
2  /home/benutzer
3  benutzer@pcname:~$ cd Documents
4  benutzer@pcname:~$ pwd
5  /home/benutzer/Documents
```

Dateityp bestimmen

Ich habe hier nun zwei Dateien “test”, von denen ich nicht weiß, was sie sind:

```
1 benutzer@pcname:~$ ls
2 test            test.mp3
```

`file`:

```
1 benutzer@pcname:~$ file test
2 test: ASCII text
3 benutzer@pcname:~$ file test.mp3
4 test.mp3: ASCII text
```

→ Text-Dateien!

Für alle Dateien und sogar Ordner, genauso wie mehrere Dateien:

```
1 benutzer@pcname:~$ file test test.mp3
2 test: ASCII text
3 test.mp3: ASCII text
```

Aufgabe

Sucht mit Hilfe von `file` 5 verschiedene Dateiarten auf eurem PC und schreibt euch die Ausgabe auf. Z.B.

- .html: XML document text
- .pdf: PDF document, version 1.5
- .tex: LaTeX 2e document, ASCII text
- .ldif: ASCII text, with CRLF line terminators
- .jpg: JPEG image data, JFIF standard 1.01
- .sh: Bourne-Again shell script, ASCII text executable

Achtung!

Bash ist “case sensitive”.

`ls` ≠ `Ls` ≠ `LS`

Nur `ls` funktioniert so, wie wir es kennen!

Beispiel:

```
benutzer@pcname:~$ LS
The program 'LS' is currently not installed. You can install it by typing:
sudo apt-get install sl
```

Zusammenfassung

`ls`	Listet alle Dateien im aktuellen Verzeichnis auf
`cd x`	Wechselt in das Verzeichnis x
	.. für Wechsel in eine Ebene höher
	kein Argument, ~ oder −− zum Homeverzeichnis
`clear`	Scrollt soweit runter, bis die aktuelle Zeile die oberste ist.
`pwd`	Print Working Directory
`file x`	Gibt den Dateityp von x aus

Hilfe!

Kann `ls` noch mehr?

```
benutzer@pcname:~$ ls --help
```

Dann passiert folgendes:

```
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them.  E.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes.  See SIZE format below.
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                              modification of file status information)
                              with -l: show ctime and sort by name
                              otherwise: sort by ctime, newest first
```

```
1   -C                         list entries by columns
2       --color[=WHEN]         colorize the output.  WHEN defaults to 'always'
3                                or can be 'never' or 'auto'.  More info below
4   -d, --directory            list directory entries instead of contents,
5                                and do not dereference symbolic links
6   -D, --dired                generate output designed for Emacs' dired mode
7   -f                         do not sort, enable -aU, disable -ls --color
8   -F, --classify             append indicator (one of */=>@|) to entries
9       --file-type            likewise, except do not append '*'
10      --format=WORD          across -x, commas -m, horizontal -x, long -l,
11                               single-column -1, verbose -l, vertical -C
12      --full-time            like -l --time-style=full-iso
13  -g                         like -l, but do not list owner
14      --group-directories-first
15                             group directories before files.
16                               augment with a --sort option, but any
17                               use of --sort=none (-U) disables grouping
18  -G, --no-group             in a long listing, don't print group names
```

...

```
  -h, --human-readable        with -l, print sizes in human readable format
                                (e.g., 1K 234M 2G)
      --si                    likewise, but use powers of 1000 not 1024
  -H, --dereference-command-line
                              follow symbolic links listed on the command line
```

...

```
SIZE may be (or may be an integer optionally followed by) one of following:
KB 1000, K 1024, MB 1000*1000, M 1024*1024, and so on for G, T, P, E, Z, Y.

Using color to distinguish file types is disabled both by default and
with --color=never.  With --color=auto, ls emits color codes only when
standard output is connected to a terminal.  The LS_COLORS environment
variable can change the settings.  Use the dircolors command to set it.

Exit status:
 0  if OK,
 1  if minor problems (e.g., cannot access subdirectory),
 2  if serious trouble (e.g., cannot access command-line argument).

Report ls bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'ls invocation'
```

Ein wenig viel, oder?
Versuche

```
benutzer@pcname:~$ ls --help |more
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them.  E.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes.  See SIZE format below.
  -B, --ignore-backups       do not list implied entries ending with ~
--More--
```

- “Enter”: Nachfolgende Zeile
- “q”: Abbruch

Analog “|less”, sogar mit Scrollen mit den Pfeiltasten.

Aufgaben

Sucht nun aus der Hilfe von `ls` 5 euch wichtig erscheinende Optionen raus (es dürfen auch gerne mehr sein) und schreibt sie euch auf!
(`ls --help |more`

Beispiele:

`-a`	do not ignore entries starting with .
`-l`	use a long listing format
`-s`	print the allocated size of each file
`-S`	sort by file size
`-t`	sort by modification time, newest first
`-X`	sort alphabetically by entry extension
`-1`	list one file per line

tab-completion

Kleine Hilfe für das Gedächtnis:
Autovervollständigung mittels Tab-Taste:

```
1 benutzer@pcname:~$ cd Doc
```

↓ Tab-Taste

```
1 benutzer@pcname:~$ cd Documents
```

Z.B. Zielverzeichnis:
Programmierungsleitungsassistenzarzthelferdokumente

Ergebnis dabei nicht eindeutig?
Benutze *Doppel*-Tab-Taste!

```
1 benutzer@pcname:~$ cd Do
```

↓ Doppel-Tab-Taste

```
1 benutzer@pcname:~$ cd Do
2 Documents          Downloads
```

Probiert es aus!

Wildcards

Wildcards = Platzhalter
Drei Arten:

?	Ein beliebiges Zeichen
??	Zwei beliebige Zeichen
...	
*	Beliebige Menge an beliebigen Zeichen
{x,y,z}	Nur wenn x,y oder z komplett vorkommen
[xyz]	Wenn einer der Buchstaben vorkommt

Beispiele:

- ls *.jpg → alle JPG-Bilder des Ordners
- ls *.{jpg,Jpg,JPG}
- ls Jahrgang_201?.jpg
- ls *.[do][od][ct] → alle .doc- und .odt-Dateien, aber auch z.B. .ddt

Abbruch

Prozess im Terminal tut nichts mehr oder muss sofort beendet werden?

“Q” hilft nicht?

Benutze `Strg+C`!

(Auch das muss nicht immer funktionieren...)

Zusammenfassung

`Befehl --help`	Zeigt die Hilfe an
`|more`	Ermöglicht ein Scrollen, falls die Ausgabe zu
`|less`	lang ist
Tab-Autovervollständigung	
Wildcards:	
?	Ein beliebiges Zeichen
*	Beliebige Menge an Zeichen
{x,y,z}	Nur wenn x,y,z vorkommen
[xyz]	Wenn einer der Buchstaben vorkommt
`Strg+C`	Bricht den Prozess ab

Aufgabe:
Entwerft einen Befehl, der euch alle Textdateien im aktuellen Verzeichnis auflistet!
Z.B. `ls *.{txt,odt,doc,docx}`

Weitere Befehle

- Bewegen mit mv
- Kopieren cp
- Löschen mit rm
- Ordner mit mkdir
- Datei öffnen mit less und cat
- Suchen mit locate und grep
- Dateien anlegen mit touch

Weiter zu “Zusammenfassung”

Weiter zu “sudo”

Bewege dich! *Zauberstab schwing*

`mv Datei Verzeichnis` (*move*)

also z.B.

```
1 benutzer@pcname:~$ mv test.mp3 Documents
```

Optionen:

-b	Legt ein Backup von jeder Zieldatei an, falls überschrieben wird
-f	Überschreibt ohne Nachfrage
-i	Überschreibt nur auf Nachfrage
-n	Kein Überschreiben
-u	Überscheibt, wenn neuer

Zurück zur "Übersicht"

Kopieren

mittels cp QUELLE ZIEL (*copy*)

```
1  benutzer@pcname:~$ cp test.mp3 Documents
2  benutzer@pcname:~$ ls Documents/
3  ...
4  test.mp3
5  ...
```

Optionen:

-n	Überschreibt keine Dateien
-b	Backup aller Zieldateien
-r	Kopiert Verzeichnisse rekursiv
-u	Update

Zurück zur "Übersicht"

Geh weg!

Datei löschen:

`rm Datei` (*remove*)

Nur für Dateien!
Verzeichnisse:

`rm -r Verzeichnis` oder empfohlener `rmdir Verzeichnis`

Achtung: Dieses Löschen ist endgültig!

Zurück zur "Übersicht"

Make it

Umgekehrt zum letzten: Erstellen von Verzeichnissen:

`mkdir Verzeichnis` (*make directory*)

Beispiele:

- `mkdir Ubucon2013`
- `mkdir Dokumente/Ubucon2013`

Zurück zur "Übersicht"

Textdateien

Beispiel: `test.mp3` war `ASCII text`.
Anzeigen:

```
less Datei oder
cat Datei
```

an.

Aufgabe:
Öffnet nun eine beliebige Textdatei einmal mit `less` und einmal mit `cat`.
Was ist der Unterschied?

- `less`: Scrollen mit Pfeiltasten, nach Beenden ist Text nicht mehr sichtbar.
- `cat`: Schreibt allen Inhalt in die Konsole.

Zurück zur "Übersicht"

Wo bist du?

Suchen einer Datei mittels

```
locate x
```

→ Wildcards!
Beispiel:
`locate *.jpg` → Zeigt alle JPG-Dateien an.
Ähnlich: `find`

- Suche nach Veränderungsdatum, Größe, Benutzerzugehörigkeit
- Auch Ordner werden gesucht
- Suche in definierter Verzeichnistiefe
- z.B. `find -empty` sucht nach Dateien und Ordnern, die leer sind

find sucht ab dem aktuellen Verzeichnis.

Zurück zur "Übersicht"

Aufgabe:

Sucht euch 5 Optionen zu "`locate`" und "`find`" aus der Hilfe!

locate:

-c	Zeigt Anzahl der Treffer an
-i	Ignoriert Groß-/Kleinschreibung
-w	Voller Name muss übereinstimmen

find:

-name Dateiname	Sucht nach Dateiname
-iname Dateiname	Sucht nach Dateiname ohne Groß-/Kleinschreibung
-mtime +/-n	Sucht nach Dateien, die jünger (-), genau (nichts) oder älter (+) als n Tage sind
-size n[cwbkMG]	Suche nach Dateigröße
-delete	Löscht gefundene Dateien

Zurück zur "Übersicht"

Suche in Dateien

Mittels

```
grep MUSTER [Datei]
```

sucht man nach einem Muster [in einer Datei].
Beispiele:

- `grep grep VortragBashWeitere.tex`
- `grep grep *.tex`

Die Angabe einer Datei ist zwar optional, empfiehlt sich aber meistens aufgrund der Suchdauer!

Optionen von `grep`:

-A xx	Gibt die xx nachfolgenden Zeilen aus
-B xx	Gibt die xx Zeilen vorher aus
-C xx	Gibt xx zusammenhängende Zeilen aus
-H	Gibt den Namen der Datei vor Treffer aus

Zurück zur "Übersicht"

Schnelles Anlegen

Schnell mal Datei ohne Inhalt:

```
touch Dateiname
```

Aber touch kann noch mehr:

-a	Modifikation der Zugriffszeit
-c	Unterdrückt das Erstellen einer Datei
-m	Modifikation der Änderungszeit

Zurück zur "Übersicht"

Zusammenfassung

`mv Datei Verzeichnis`	Verschiebt die Datei in Verzeichnis
`rm Datei`	Löscht die Datei
`rm -r Verzeichnis`	Löscht das Verzeichnis
`mkdir Verzeichnis`	Legt ein Verzeichnis an
`less Datei`, `cat Datei`	Zeigt den Inhalt einer Textdatei an
`locate x`	Sucht nach einem Dateinamen x
`find`	Sucht nach Dateien mit vielen Optionen
`touch Datei`	Ändert Attribute der Datei oder legt sie an

sudo - substitute user do

- Befehl, um ein Programm im Namen eines anderen Benutzers zu starten

→ Administratorrechte

- Z.B. `sudo apt-get install firefox`

Aber Achtung:

- `$HOME` wird nicht geändert.
- Für grafische Anwendungen besser: `gksudo` (Unity, GNOME, Xfce, LXDE), `kdesudo` (KDE), `sudo -H` ("alle")

Zusammenfassung

Terminalprogramm	
sudo	führt Befehl als root aus
sudo -H	führt Befehl als root aus mit angepasstem $Home
sudo -i	startet root-Shell
sudo -u Benutzer	startet Befehl als Benutzer

Grafisches Programm	
kdesudo/gksudo	führt grafisches Programm als root aus
kdesudo -u Benutzer	startet grafisches Programm als Benutzer
gksu -w -u	wechselt Benutzer mit Passwort

apt-get

Befehl: `apt-get install [OPTION]`

Option	Bedeutung
`install`	Installiert das angegebene Paket
`remove`	Entfernt das angegebene Paket
`purge`	Entfernt das angegebene Paket sowie alle Einstellungen
`update`	Aktualisiert die Paketquellen
`upgrade`	Installiert neue Paketversionen
`autoremove`	Entfernt ungenutzte Pakete
`-f install`	Repariert defekte Pakete

Suche nach Paketen: `apt-cache search GESUCHTES`
Informationen zu Paketen: `apt-cache show PAKETNAME`
Alle Pakete bekommt man mittels `dpkg -l`.

Paketquelle hinzufügen

mit

```
sudo add-apt-repository PAKETQUELLE.
```

Anschließend

```
sudo apt-get update
```

und

```
sudo apt-get install PAKET
```

ACHTUNG: Fremdpakete können das System gefährden. Für genauere Informationen siehe: http://wiki.ubuntuusers.de/Fremdquellen.

Manuelle Installation

Erfolgt in 3 bis 4 Schritten:
Terminal im entsprechenden Ordner öffnen und dann

1. `./configure`
2. `make`
3. (`make test`)
4. `sudo make install`

Hinweis: `make test` schlägt manchmal fehl, obwohl alles funktioniert!

Ausschalten

Herunterfahren:

- sofort: `sudo shutdown -h now`
- in xx Minuten: `sudo shutdown -h +xx`
- Abbruch mit Ctrl-C oder `sudo shutdown -c`

Neustart: `sudo reboot`

Welches System?

```
1  benutzer@pcname:~$ uname -a
2  Linux PCName 3.8.0-30-generic #44-Ubuntu SMP Thu Aug 22
3  20:54:42 UTC 2013 i686 athlon i686 GNU/Linux
```

⇒ Ausgabe des Kernels

Meist passender:

```
1  benutzer@pcname:~$ lsb_release -a
2  No LSB modules are available.
3  Distributor ID: Ubuntu
4  Description:    Ubuntu 13.04
5  Release:        13.04
6  Codename:       raring
```

⇒ Versionsnummer und Name

Listen

- Liste Hardware: `lshw` (*list hardware*)
- Liste USB-Geräte: `lsusb` (*list USB*)
- Liste PCI-Steckkarten: `lspci` (*list PCI*)

Beispiel:

```
benutzer@pcname:~$ lsusb
Bus 004 Device 002: ID 046d:c30e Logitech, Inc. UltraX Keyboard (Y-BL49)
Bus 004 Device 003: ID 1241:1166 Belkin MI-2150 Trust Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Umleitungen

Auflistungen können sehr lang werden!
Lösung: Umleitungen

`> Datei`	Standardausgabe umleiten oder
`>> Datei`	... an Datei anhängen
`< Datei`	Eingabe aus Datei holen
`| Befehl`	an Befehl weitergeben

Beispiele:

- `dpkg -l > Paketliste.txt`
- `ls > Dateiliste.txt`

Aufgabe

1. Lasst euch eine Hardwareliste in eine Datei ausschreiben!
2. Was macht dieser Befehl?
   `dpkg -l | grep libre`

Hinweise:

`lshw`	Liste Hardware
`lsusb`	Liste USB-Geräte
`lspci`	Liste PCI-Steckkarten
`> Datei`	Standardausgabe umleiten oder
`>> Datei`	... an Datei anhängen
`< Datei`	Eingabe aus Datei holen
`| Befehl`	an Befehl weitergeben
`grep BEGRIFF [DATEI]`	Sucht in DATEI nach BEGRIFF

Entpacken

Erstellen	`rar a Archivname Dateien...`
Entpacken	`unrar x Archivname.rar`
tar.gz erstellen	`tar -czf Archivname.tar.gz Dateien...`
tar.gz entpacken	`tar -xzf Archivname.tar.gz`
Für tar.bz2 ersetze “zf” durch “jf”.	
.zip entpacken	`unzip Archivname.zip`

Zusatz

- Dateiverwaltung: chmod, stat, dd, rename
- Internet: wget, Webbrowser
- Dateisysteme: mount, unmount, blkid
- Netzwerk: ifconfig, ping, traceroute, dig
- System: ps, kill, top, free, df

chmod - change mod

Ändert die Zugriffsrechte einer Datei.
Befehl: `chmod [OPTION] MODUS DATEI`

r	Leserechte (read)	u	Besitzer
w	Schreibrechte (write)	g	Gruppe
x	Ausführrechte (execute)	o	Andere Benutzer
+	Fügt hinzu	-	Entfernt

Beispiel:

- `chmod ugo+rwx DATEI`: Gibt allen Benutzern Lese-, Schreib- und Ausführrechte
- `chmod +x MeineBashDatei.sh`: Erlaubt allen Benutzern, die Datei auszuführen
- `chmod o=r Liste.txt`: Erlaubt anderen Benutzern nur, die Datei zu lesen

stat

Gibt eine Statistik für Größe, Ort, Zugriffsrechte, Zugriffszeit etc. für eine Datei aus.
Befehl: `stat DATEI`

dd

Mit *dd* kann man Dateien bitgenau kopieren. Damit lässt sich z.B. die ISO eines Live-Systems bootfähig auf einen USB-Stick schreiben oder eine exakte Kopie des eigenen Systems erstellen.
Befehl: `dd if=QUELLE of=ZIEL [OPTIONEN]`

- Festplatte klonen: `dd if=/dev/sda of=/dev/sdb`
  Ziel sollte mindestens genauso groß sein, wie die Quelle!
- Partition klonen: `dd if=/dev/sda1 of=/dev/sdb1`
- Festplatte überschreiben:
  `dd if=/dev/zero of=/dev/sda` (schnell)
  `dd if=/dev/urandom of=/dev/sda` (langsam aber sicherer)
  Vorsicht damit!
- Live USB-Stick erstellen:
  1. USB-Stick aushängen
  2. `sudo dd if=hybrid_iso_image.iso of=/dev/sdc bs=4M`
  3. `sync`

  Es ist auf ein Hybrid-Image zu achten (Ubuntu's Images sind dies ab 11.10)

rename

Umbenennen einer Datei.
Befehl: `rename [OPTION] 'REGULAERER_AUSDRUCK' DATEI(EN)`
Optionen:

-n	Ändert nichts, zeigt nur das theoretische Ergebnis an
-f	Überschreibt Datei, falls der Name schon vorhanden ist
-v	Schreibt Erfolgsnachricht ins Terminal

Für regläre Ausdrücke siehe
http://wiki.ubuntuusers.de/rename#Syntax-der-regulaeren-Ausdruecke-in-Perl

Downloaden

Befehl: `wget [Optionen] URLs`

`-i Datei`	URLs aus einer Text- oder HTML-Datei auslesen
`-t [x]`	x Versuche, bis abgebrochen wird
`-c`	Versucht einen abgebrochenen Download fortzusetzen
`-N`	Download nur, wenn Datei neuer als vorhandene
`-k`	Konvertiert externe Links in interne
`--limit-rate=Y`	Begrenzt die Downloadrate auf X
`-b`	Download im Hintergrund
`--user=USER`	Angabe des Benutzers und
`--password=PSWD`	des Passwortes für z.B. ftp

Webbrowser - w3m

Installation: `sudo apt-get install w3m w3m-img`
Aufruf: `w3m URL`
Navigation mit den Pfeiltasten, Enter wählt einen Hyperlink oder ein Textfeld für Eingaben aus.
Kürzel:

Umschalt+H	Hilfsdatei für Tastenkürzel	Q	Schließt w3m
Tab (+Esc)	Springt zum nächsten (vorherigen) Link	Esc+L	Öffnet eine Liste an Links der Seite
Umschalt+U	Internetadresse eingeben	Esc+S	Speichert die Seite ab

Webbrowser - Lynx

Installation: `sudo apt-get install lynx-cur`
Aufruf: `lynx URL`
HTML zu ASCII-Text: `lynx -dump quelle.html > ziel.txt`
Kürzel:

K	Zeigt Tastaturkürzel an	Enter/→	Link folgen
Return	Liste der bisher angezeigten Seiten	←	Zurück
Bild↑/↓	Scrollen	Q	Beendet Lynx
↑/↓	Zum nächsten/vorherigen Link		

Webbrowser - links2

Installation: `sudo apt-get install links2 gpm`
Aufruf: `links2 (-g) URL` (-g für Grafikmodus)
Kürzel:

Esc	Menü	Q	Beendet Links2
Bild↑/↓	Rauf/Runterscrollen	Umschalt+G	Gehe zu Adresse
→	Link folgen	←	Link zurück

Dateisysteme

Befehle:

`mount -l`	Ausführliche Anzeige aller eingehängter Dateisysteme
`mount [OPTION] Gerät`	Einhängen/Aushängen von Dateisystemen
`sudo umount`	
`blkid`	Anzeige aller Dateisysteme mit UUID

- Partitionen werden in der Datei `/etc/fstab` verzeichnet
- Statisches Einbinden: Partition wird beim Systemstart eingebunden
- Tempöräres Einbinden: Partitionen, die beim Betrieb ausgewechselt werden können, z.B. DVDs oder USB-Sticks
- Oft werden “sudo”-Rechte benötigt

mount - Optionen

-a	Hängt alle Geräte in `/etc/fstab` ein
-f	Simuliert nur das Einhängen
-o Option, nämlich z.B.:	
defaults	Standardoptionen: rw, suid, dev, exec, auto, nouser, async
sync	Puffer wird sofort geschrieben
async	Puffer wird nicht sofort geschrieben
nouser bzw. user	Nur root bzw. jeder kann ein-/aushängen
(no)auto	(Nicht) automatisch beim Systemstart eingehängen
rw bzw. ro	Einhängen read-write bzw. read-only
dev	Gerätedateuen werden interpretiert
exec	Binärdateien können ausgeführt werden
suid	Programme mit SetUID- oder SetGID-Bit können ausgeführt werden
remount	Aus- und Einhängen

ntfs-Partitionen

Tempörär:

```
sudo mount -t ntfs -o uid=1000,umask=0022 /dev/sdaX /media/NAME
```

Statisch:

Eintrag in */etc/fstab*:

```
/dev/sdaX /media/NAME ntfs uid=1000,umask=0022 0 0
```

Hierbei entspricht *sdaX* der ersten Bezeichnung aus *blkid* und "NAME" dem Namen, der vergeben werden soll. Unter Umständen muss ein Einhängepunkt erst erstellt werden via *sudo mkdir /media/NAME*.

ifconfig und ping

ifconfig
Zeigt alle vorhandenen Netzwerk-Karten mit MAC-Adresse etc. an.
Befehl: `ifconfig`

ping
Überprüft, ob ein Ziel im Netzwerk/Internet erreicht werden kann.
Befehl: `ping ZIEL`
Beispiel:

- `ping ubuntuusers.de`
- `ping 127.0.0.1`

traceroute und dig

traceroute Verfolgt die "Route, die eine Webseite zurücklegt, bis sie auf dem Bildschirm erscheint."
Befehl: `traceroute ZIEL`
Beispiel: `traceroute www.ubuntuusers.de`

dig
Steht für "Domain information grouper" und fragt Informationen vom DNS-Server ab.
Befehl: `dig ZIEL`
Beispiel: `dig www.ubuntuusers.de`

ps - processes
Schreibt alle aktuellen Prozesse mit Prozess Identification Number (PID) aus.
Befehl: `ps -A`

kill
Beendet einen Prozess, kennt dabei sehr viele Optionen. Am einfachsten bedient man *kill* über die PIDs.
Befehl: `kill PID`

top
Zeigt alle aktuellen Prozesse (u.a. mit PID) und Systemressourcen dynamisch an, entspricht also dem Task-Manager unter Windows. Beenden mit “q”.
Befehl: `top`

free

Zeigt die momentane Speicherbelegung des RAM an.
Befehl: `free`
Optionen:

-b/-k/-m	Anzeige in Byte/Kilobyte/Megabyte
-t	Anzeige einer Total-Zeile

df - disk free

Schreibt eine Liste aller vorhandener Dateisysteme mit Speichernutzung aus.
Befehl: `df [OPTION]`
Optionen:

-a	Alle Dateisysteme
-h	Angenehmere Anzeige der Dateisystemgröße
-l	Nur lokale Partitionen

Shell-Programmierung

Anfang eines Shell-Skriptes mit Endung .sh: `#!/bin/bash` (Genannt "Shebang")
Aufruf im Terminal: `bash NameDesSkriptes.sh`
Der Name sollte kein vorhandener Befehl sein; zum Testen kann man `type NameDesSkriptes` verwenden.
In Bash-Skripten können natürlich sämtliche Befehle verwendet werden, die auch ins Terminal eingetippt werden können, z.B. `ls`, `rm`, aber auch Umleitungen (<,>,»,|).
Hallo World!

```
#!/bin/bash
echo "Hello World!"
```

```
1 ~$ bash HelloWorld.sh
2 Hello World!
```

Variablen

werden definiert mit: `VariablenName=Inhalt`
und aufgerufen mit: `$VariablenName`
Variable löschen: `unset VariablenName`

```
name="Mein Name ist Max Mustername."
echo "$name"
```

```
1 Mein Name ist Max Mustername.
```

Arrays

werden definiert mit

- `ArrayName=(Inhalt1 Inhalt2 Inhalt3 ...)`
- `ArrayName[i]=Inhalt_i`

und aufgerufen

- einzeln: `${ArrayName[i]}`
- komplett als *ein* Argument mit: `${ArrayName[*]}`
- als *verschiedene* Argumente mit : `${ArrayName[@]}`

Anfügen: `ArrayName+=(Das hier kommt dazu)`

Array löschen: `unset ArrayName`

Länge eines Arrays mit: `count=${#ArrayName[@]}`

Interaktion

Mit folgender Zeile fragt man die Eingabe einer Zeichenkette ab:

```
read -p "Bitte Antwort eingeben" Antwort
echo "$Antwort"
```

Entscheidungen: If-Else

Syntax:

```
if [ Bedingung ]
 then
   Befehle
 elif [ Bedingung2 ]
   then
     Befehle
 else
   Befehle
fi
```

Natürlich muss kein `elif` oder `else` vorkommen.

Entscheidungen: Bedingungen

Für die Bedingungen eignen sich

-d Verzeichnis	Wahr, wenn Verzeichnis existiert
-e Datei	Wahr, wenn Datei existiert
-n String	Wahr, wenn String nicht leer ist
x = y	Vergleich zweier Strings
x -eq y	Vergleich zweier Zahlen
x -lt y	less than
x -gt y	greater than
x -le y	less or equal
x -ge y	greater or equal
x -ne y	not equal
! x	negiert Aussage x

Entscheidungen: Case

Prüft alle Fälle ab. Bei vielen Möglichkeiten besser als if-else

```
case "$variable" in
 Fall1)
   Befehle
 ;;
 Fall2)
   Befehle
 ;;
 ...
esac
```

Wiederholungen: For-Schleife

Wiederholt solange richtig.

```
for i in "Parameterliste"
 do
   Befehle
done
```

Hierbei kann die Parameterliste z.B. auch ein Array `${ArrayName[@]}` sein.

Parameter übergeben und Funktionen

Parameter werden mit den Variablen $1, $2,... angesprochen.

```
echo "$1"
```

```
1 ~$ NameDesSkriptes.sh Hallo
2 Hallo
```

Funktionen

```
Funktionsname(){
   Befehle
}
```

Hierbei können natürlich auch Parameter übergeben und mit $1 etc. innerhalb angesprochen werden.

Literatur für Shell-Programmierung

- http://wiki.ubuntuusers.de/Shell/Bash-Skripting-Guide_für_Anfänger
- http://openbook.galileocomputing.de/shell_programmierung/

Quellen

- ubuntuusers.de: http://ubuntuusers.de/
  - http://wiki.ubuntuusers.de/Shell
  - http://wiki.ubuntuusers.de/Shell/Befehlsübersicht
  - http://wiki.ubuntuusers.de/Bash
  - http://wiki.ubuntuusers.de/Bash/Prompt
- Erste Schritte mit Ubuntu 10.04 - Zweite Ausgabe.pdf: http://ubuntu-manual.org/Ubuntu Manual Project, hierbei speziell http://ubuntu-manual.org/download/10.04e2/de/screen, ab Seite 125ff
- Linux auf einem Blatt http://helmbold.de/artikel/Linux-auf-einem-Blatt.pdf
- Shell-Übersicht http://www.321tux.de/wp-content/uploads/2010/03/shell-uebersicht.pdf
- Shell-Workshop http://www.pro-linux.de/artikel/1/44/shell.html
- Advanced Bash-Scripting Guide http://tldp.org/LDP/abs/html/