UNIX-Referenzkarte

Verzeichnis . . .

aktuelles anzeigen	`pwd`
wechsle nach . . . (`$HOME`)	`cd` [⟨Zielverzeichnis⟩]
erzeugen	`mkdir` ⟨Verzeichnis⟩
leeres löschen	`rmdir` ⟨Verzeichnis⟩
umbenennen	`mv` ⟨vorher⟩ ⟨nachher⟩
kummulierte Größe	`du -hs` [⟨Verzeichnis oder Datei⟩]
	`-L` symbolische Links dereferenzieren

Datei(en) . . . (Elementares)

anzeigen	`less` ⟨Datei(en)⟩
	`b` `f` Seite hoch/runter
	`g` `G` zum Anfang/Ende springen
darin	`/` `?` Suchen vor-/rückwärts
	`n` Suche wiederholen
	`q` beenden
archivieren	`tar` ⟨Datei(en)⟩
	`-c` erzeugt neues Archiv
	`-f` benutze angegebene Datei als Archiv
	`-p` archiviert auch Zugriffsrechte
	`-r` füge Datei(en) hinten an Archiv an
	`-t` Archivinhalt listen
	`-x` aus dem Archiv extrahieren
	`-z` komprimiert Archiv mit `gzip`
auflisten	`ls` ⟨Datei(en)⟩|⟨Verzeichnis⟩
	`-a` alle (auch `.*`)
	`-d` Verzeichnisnamen statt -inhalten
	`-F` Typ kennzeichnen (`/*@`)
	`-l` langes Format
ausgeben	`cat` ⟨Datei(en)⟩
	`-n` Zeilen nummerieren
	`head` bzw. `tail`
	`-`*n* erste bzw. letzte *n* Zeilen
dekomprimieren	`bunzip2` ⟨Datei(en)⟩
	`-f` beim Auspacken überschreiben
	`gunzip` ⟨Datei(en)⟩
	`unzip` ⟨ZipDatei⟩ [⟨Datei(en)⟩]
drucken (Sys-V)	`lp -d` ⟨Drucker⟩ ⟨Datei(en)⟩
	abbrechen: `cancel` ⟨Job-ID⟩
durchsuchen	`grep` ⟨Ausdruck⟩ ⟨Datei(en)⟩
	`-l` nur Dateinamen anzeigen
editieren	(x)`emacs, nedit, pico, nano, vi`(`m`)`, me`
komprimieren	`bzip2` ⟨Datei(en)⟩
	`-t` Integrität der komprimierten Datei prüfen
	`-z` komprimieren
	`gzip` ⟨Datei⟩
	`zip` ⟨ZipDatei⟩ ⟨Datei(en)⟩
kopieren	`cp` ⟨Quelle(n)⟩ ⟨Ziel⟩
	`-r` rekursiv (d.h. auch Verzeichnisbäume)
löschen	`rm` ⟨Datei(en)⟩
	`-f` keine Sicherheitsrückfragen
	`-r` rekursiv (d.h. auch Verzeichnisbäume)
Typ feststellen	`file` ⟨Datei⟩
vergleichen	`cmp` (keine Aussage über Qualität der Unterschiede)
verschieben	`mv` ⟨Quelle(n)⟩ ⟨Ziel(verzeichnis)⟩

Datei(en) . . . (Fortgeschrittenes)

abzählen	`wc` ⟨Datei(en)⟩
	`-c` Zeichen zählen
	`-l` Zeilen zählen
	`-w` Wörter zählen
Duplikatzeilen	`uniq` ⟨Eingabedatei⟩ [⟨Ausgabedatei⟩]
herunterladen	`wget` ⟨URL⟩ (HTTP oder FTP)
	`-c` fortsetzen
patchen	`patch <` ⟨Patchdatei⟩
	`-p`*k* entferne *k* führende `/` in Pfadangaben
Prüfsumme	`md5sum` ⟨Datei(en)⟩
	`-c` Liste in Summendatei überprüfen
Spalten extrah.	`cut -c`*n* ⟨Datei(en)⟩ (Zeichen-Bereichsangabe *n*)
oder	`-f`*m* Feld-Bereichsangabe *m*
	`-d`*t* Trennzeichen *t* (Standard: Tabulator)
suchen	`find` ⟨Verzeichnis(se)⟩ ⟨Bedingung(en)⟩
	`-atime` *n* letzter Zugriff vor *n* Tagen?
	`-exec` *k* `\;` Kommando *k* erfolgreich? (`{}`)
Bedingungen	`-name` sucht nach Namen (-smuster)
	`-print` gibt gefundene Dateien aus
	`-size` sucht nach Größe ([`+-`]⟨n⟩`c`)
	`-type` sucht nach Typ (`f,d,l`)
transferieren	`sftp`
	`scp` [⟨User⟩`@`⟨Host⟩`:`]⟨Quelle(n)⟩ [⟨User⟩`@`⟨Host⟩`:`]⟨Ziel⟩
	`-r` rekursiv
	`-C` Kompression verwenden
vergleichen	`diff` ⟨Original⟩ ⟨geänderte Datei⟩ [> ⟨Patchdatei⟩]
	`-r` rekursiv zwei Verzeichnisse vergleichen
	`-u` „unified format“ verwenden (empfohlen)
verlinken	`ln -s` ⟨Ziel(e)⟩ ⟨Link(Verzeichnis)⟩
	⟨Link⟩ `->` ⟨Ziel⟩, Ziele nur mit Verzeichnis
zerteilen	`split` ⟨Datei⟩ ⟨Präfix der Ausgabedateien⟩
	`-`*n* nach je *n* Zeilen
	`-b`*m* nach je *m* Bytes

PostScript- (PS) und PDF-Dateien

PS	anzeigen und drucken	`gv`
	nach PDF wandeln	`ps2pdf` ⟨PS-Datei⟩ [⟨PDF-Datei⟩]
	n Seiten verkleinert anordnen	`psnup -`*n*⟨groß⟩ [⟨klein⟩]
EPS	nach PDF wandeln	`epstopdf`
PDF	anzeigen und drucken	`acroread, xpdf` oder `gv`
	nach PS wandeln	`pdf2ps` (oder in Datei drucken)

Rechte ändern

`chmod` ⟨Wem⟩⟨Wie⟩⟨Welche⟩ ⟨Datei(en) und Verzeichnis(se)⟩

`-R` rekursiv für alle Dateien **und** Verzeichnisse

⟨Wem⟩: `u` Eigentümer, `g` Gruppe, `o` Sonstige, `a` Alle zusammen

⟨Wie⟩: `+` gewähren, `-` entziehen, `=` exklusiv setzen

⟨Welche⟩:

	bei Datei	*bei Verzeichnis*
`r` (4)	Lesen	`ls`, Durchsuchen
`w` (2)	Schreiben	Dateien anlegen und löschen
`x` (1)	Ausführen	`cd`, Zugriff auf Dateien
`X`	undefiniert	`x`
`s`	`u`-/`g`-ID setzen	neue Datei erhält Verz.-`g`-ruppe
`t`	undefiniert	nur Besitzer darf Datei löschen

`u/g/o` Rechte wie die von `u/g/o` setzen

Zeichenstrom . . .

abzweigen in Datei	`tee` ⟨Datei(en)⟩
	`-a` anhängen statt überschreiben
als Befehlsargument	`xargs` ⟨Optionen⟩ ⟨Kommando mit Argmt.⟩
	`-i`*m* Aufruf mit jeder Zeile als *m* (`{}`)
	`-l`*n* je *n* Zeilen verwenden
	`-p` vor Ausführung rückfragen
	`-t` fertige Kommandozeilen ausgeben
durchsuchen	`grep` ⟨Ausdruck⟩ ⟨Datei(en)⟩
	`-f` Ausdrücke aus angeg. Datei suchen
	`-i` Groß-/Kleinschreibung ignorieren
	`-v` gibt Zeilen aus, die Ausdr. nicht enth.
	`-x` Vergleich mit ganzer Zeile
seitenweise ausgeben	`less`
sortieren	`sort`
	`-n` numerisch
Zeichen ersetzen (1:1)	`tr` ⟨Wort1⟩ ⟨Wort2⟩
oder:	`-c` bilde ASCII-Komplement von Wort1
	`-d` lösche alle Zeichen aus Wort1

Netz

Remote-Login	`ssh` ⟨Benutzername⟩`@`⟨Rechnername⟩ [⟨Befehl⟩]
	`-C` Kompression verwenden
	`-X` X-Weiterleitung aktivieren
WWW	`firefox` | `mozilla` | `opera` | `lynx` | `links` | `w3m`

Dateien via FTP übertragen	`ftp` | `ncftp` | `lftp` | `yafc`
Ist Rechner erreichbar?	`ping` ⟨Rechnername⟩

X

Terminal-Emulation	`xterm`	
Fenster-Ressource löschen	`xkill`	(dann hineinklicken)
Bildschirm sperren	`xlock`	(dann Passwort eingeben)
Rechner	`xcalc`	
Manual	`xman`	

Verschiedenes

Hilfeseite anzeigen	`man` ⟨Befehl⟩|⟨Thema⟩
Zeit ausgeben	`date`
Kalender	`cal` [[⟨Monat⟩] ⟨Jahr⟩]
angemeldete Benutzer?	`who` oder `w`
Benutzerdaten anzeigen	`finger` ⟨Benutzername⟩
Identität ändern	`su` [`-`] ⟨Benutzername⟩
Passwort ändern	`passwd` [⟨Benutzername⟩]
Terminal-Mitschrift	`script` (abschließend `C-d`)
Terminal-Multiplexer	`screen` [`-r`]

Kommandos: `C-a`

`a`	sende `C-a` (für Emacs)
`C-a`	zuvor benutztes Fenster
`c`	neues Fenster (mit Shell) anlegen
`d`	`screen` von Fenstern abklemmen
`H`	Aufzeichnung d. Ausgabe ein/aus
`n`/`p`	nächstes/letztes Fenster
`"`	Fensterliste anzeigen

Bash-Referenzkarte

Begriffe

Metazeichen Leerzeichen oder `|&;()<>`; besitzt Sonderbedeutung, die durch Quoting (s.re.) lokal unterdrückt (maskiert) werden kann

Bezeichner Folge von Buchstaben, Unterstrichen und Ziffern

Wort Kette von Zeichen ohne Metazeichen

Trennzeichen Leerzeichen, Tabulator

Kommando einfaches K., geklammertes K. (`(␣<Subshell-K.>)` oder `{␣<K.>;}`) oder Programmiersprachen-Konstrukt

Einfache Kommandos

haben einen **exit-Status** (Rückgabewert `$?`) gleich 0 genau dann, wenn sie erfolgreich ausgeführt wurden. (Dies ist genau invers zur Konvention für C-Funktionen.) Beispiele: alle Datei- oder Verzeichnis-Befehle.

Pipelines

sind durch von Pipes `|` getrennten Kommandos. Die Standardausgabe des Kommandos links der Pipe wird als Standardeingabe an das rechts stehende Kommando weitergegeben. Der exit-Status wird allein vom letzten Kommando bestimmt.

(Kommando-) Listen

sind Folgen von Pipelines, jeweils getrennt durch:

`;`	bewirkt sequentielle Ausführung
`&`	bewirkt parallele Ausführung
`&&`	Ausführung nur nach Erfolg (exit-Status 0) vorheriger Pipelines
`||`	Ausführung nur nach Misserfolg vorheriger Pipelines

Der exit-Status wird allein vom zuletzt ausgeführten Kommando bestimmt.

Wildcards in Dateinamen

`?`	ein beliebiges Zeichen
`*`	0 oder mehr beliebige Zeichen
`[ ]`	eines der Zeichen zwischen den Klammern
`[! ]`	ein Zeichen, welches *nicht* in den Klammern steht
`{ }`	eines der durch Komma getrennten Worte

`?` und `*` können nicht für Punkte an der ersten Stelle stehen. Ebenso muss `/.` oder `/` am Anfang des Wortes explizit angegeben werden.

Kommandosubstitution

ermöglicht, die Standardausgabe eines Kommandos *k* als Teil eines neuen Kommandos zu verwenden: `` `k` `` oder `$(k)`. Alle Metazeichen in *k* behalten ihre Sonderbedeutung. (Bem.: `` `< ⟨datei⟩` `` ist effizienter als `` `cat ⟨datei⟩` ``)

Quoting

	"	'	`	\	$	*?[	⟨CR⟩
"	e	j	n	n	n	j	z
'	j	e	j	j	j	j	z
`	n	n	e	n	n	n	n
\	j	j	j	j	j	j	z

Maskiert das Zeilenzeichen das Spaltenzeichen?
j ja, **n** nein, **e** nein, denn dort endet die Maskierung, **z** ja, aber der Zeilenvorschub bleibt erhalten

Prozesse und Jobs ...

im Hintergrund starten	⟨Befehl⟩ `&`
Jobs anzeigen (gleiche Shell)	`jobs`
	`-l` PIDs mitausgeben
	`-x` Befehl ausführen, z.B. `kill` ⟨Job⟩
Jobs addressieren	`%`⟨Jobnummer⟩
	`%`⟨Namensanfang⟩
	`%?`⟨Teilstring des Namens⟩
anhalten (im Vordergrund)	`C-z` (oft `Strg-Z`)
wieder in den Vordergrund	`fg` [⟨Job⟩] (oder ⟨Job⟩)
oder in den Hintergrund	`bg` [⟨Job⟩] (oder ⟨Job⟩`&`)
niedriger priorisiert starten	`nice` -⟨Erhöhung⟩ ⟨Befehl⟩
bei Ausloggen nicht beenden	`nohup` ⟨Befehl⟩
Prozesse anzeigen	`ps`
	`-e` alle
	`-f` ausführliche Informationen
	`-u` alle eines Benutzers
Prozess in Baumstruktur	`pstree`
Ressourcen-Rangliste	`top`
Signal senden	`kill` ⟨PID⟩
	`-9` sofort Beenden (`KILL`)
	`-15` default: beenden (`TERM`)
	`-18` weiterlaufen lassen (`CONT`)
	`-19` anhalten (`STOP`)

Shell-Skripte

Erste Zeile enthält Pfad zur Shell: `#!/bin/bash`

Ausführung in Subshell	`bash` ⟨name⟩ oder ⟨name⟩ (falls ausführbar)
direkte Ausführung	`.␣`⟨Datei⟩ oder `source` ⟨Datei⟩

Shelloptionen

werden in Shell-Skripten zur Fehlersuche eingesetzt. Dazu gibt man sie als Optionen der Shell in der ersten Zeile des Skriptes an:

`-e`	bei Fehler sofort beenden
`-n`	nur Syntax prüfen, nichts ausführen
`-v`	ausführlich arbeiten
`-x`	Kommandos vor Ausführung anzeigen

Im interaktiven Betrieb: setzen `set -`⟨Option⟩; löschen `set +`⟨Option⟩

Positionsparameter

beinhalten die an ein Shell-Skript übergebenen Argumente. Zugriff auf ihren Wert durch `$1, ..., $9, ${10},...`, Verschiebung aller Werte auf nächstkleineren Positionsparameter durch `shift`.

Shell-Variablen

Eine Shell-Variable hat einen Namen (Bezeichner) und einen Wert. Wird sie zusätzlich `export`iert, dann nennt man sie Umgebungsvariable. Auf den Wert von ⟨var⟩ greift man durch `$`⟨var⟩ oder `${`⟨var⟩`}` zu.

Shellvariablen listen	`set`
Umgebungsvariablen listen	`env`
Shellvariable setzen	⟨var⟩=⟨Wert⟩
Umgebungsvariablen setzen	`export` ⟨var⟩[=⟨Wert⟩]
Einlesen von der Standardeingabe	`read` ⟨var1⟩ ⟨var2⟩ ...

Manche Shell-Variablen werden von der Shell mit einem Wert belegt (z.B.: `HOME, PATH, PS1`), einige kann man sogar nur lesen (automatische Variablen):

`#`	Anzahl gesetzter Positionsparameter
`?`	exit-Status des zuletzt ausgeführten Vordergrundkommandos
`*`	alle Positionsparameter als **ein** String
`@`	alle Positionsparameter als **einzelne** Strings

Programmiersprachen-Konstrukte

KL steht für Kommandoliste (s.o.). Statt einem Zeilenumbruch kann auch `;` stehen.

```
if <KL als Bedingung>     while|until <KL als Bedingung>
    then <KL>                 do <KL>
fi                        done
```

Nützlich als Bedingung ist `test` ⟨Bedingung⟩ (auch: `[` ⟨Bedingung⟩ `]`). Der Rumpf wird nur ausgeführt, falls die KL erfolgreich abgelaufen ist (Testkriterium erfüllt, exit-Status 0).

```
for <Variable> in <Worte>   case <Wort> in
    do <KL>                     <Muster1>) <KL1>;;
done                            <Muster2>) <KL2>;;
                            ...
                                <Mustern>) <KLn>;;
                            esac
```

Funktionen

Definition: `<Funktionsname>()␣{␣<KL>;}` (Zugriff auf Argumente via `$1, $2`)
Aufruf: `<funktionsname>` [`<Argument(e)>`]

Ein-/Ausgabeumleitung

Eingabe aus Datei	⟨Befehl⟩ < ⟨Datei⟩
Ausgabe in Datei (löschen)	⟨Befehl⟩ > ⟨Datei⟩
Ausgabe in Datei (anfügen)	⟨Befehl⟩ >> ⟨Datei⟩
Eingabe aus Skript	⟨Befehl⟩ <<'⟨EOF⟩'
Fehlerausgabe in Datei	⟨Befehl⟩ 2>[>] ⟨Datei⟩

Sonstiges

Alias definieren	`alias` ⟨Name⟩=⟨Kommando⟩
Befehl finden	`type` ⟨Befehl⟩
Terminal leeren	`clear` oder `C-l`