bash – Shellbefehle (alphabetisch)

Syntax in der folgenden Befehlsbeschreibung:

- Großbuchstaben + Unterstreichung: Platzhalter für einen beliebigen String (Beispiel: NAME)
- Eckige Klammern: In den Klammern steht eine optionale Ergänzung, die weggelassen werden kann (Beispiel: [-r])

Strg-c	Laufendes Programm abbrechen
Strg-d	Laufende Eingabe schließen (Shell beenden)
alias NEW=OLD	Zeichenersetzung von String OLD durch String NEW definieren (ohne Leerzeichen!)
apropos STRING	Sucht den STRING in allen Hilfetexten und gibt die gefundenen Befehle aus
cat FILE	Textdatei FILE auf Standardausgabe (stdout) ausgeben
cd [DIR]	Aktuelles Arbeitsverzeichnis zu DIR wechseln
chmod [ugoa][+/-][rwx] FILE	Zugriffsrechte bzgl. Datei FILE ändern (x: ausführbar) *Bsp.: chmod a+r myfile → Alle erhalten Leserecht für myfile*
cp [–i] SOURCE DEST	Kopiere die Datei SOURCE. Die neue Dateikopie heißt DEST (im selben Verzeichnis) oder liegt im Verzeichnis DEST unter dem Namen SOURCE *(wenn DEST ein Verzeichnis ist)*
date	Datum und Zeit anzeigen
df [-h]	Informationen über Dateisysteme anzeigen
diff FILE1 FILE2	Unterschiede zwischen Datei FILE1 und Datei FILE2 anzeigen
du [-h –d LEVEL] [DIR]	Platzverbrauch für Verzeichnisse anzeigen, beginnend bei DIR oder im aktuellen Verzeichnis
echo STRING	Zeichenkette STRING auf Standardausgabe (stdout) ausgeben
env	Umgebungsvariablen anzeigen
exit	Shell oder Skript beenden
export VAR	Shell-Variable VAR an alle Kindprozesse vererben
find DIR -name “FILE” -print	Suche in allen Verzeichnissen und Unterverzeichnissen (beginnend im Verzeichnis DIR) eine Datei namens FILE und gib den Dateipfad aus
grep [-r] STRING [FILE]	Suche in der Datei FILE (oder stdin) nach der Zeichenkette STRING, ggf. rekursiv in allen Unterverzeichnissen
head [-n] FILE	Die ersten n Zeilen der Textdatei FILE ausgeben
jobs	Information über Hintergrund-Programme der aktuellen Shell
kill [–9] PID	Prozess mit der Prozess-ID PID abbrechen (beenden)
locate STRING	Finde alle Dateien, in deren Namen die Zeichenkette STRING vorkommt und gib den Dateipfad aus (Achtung: Suche in einer Datenbank! Aktualisierung mit updatedb)
ln [-s] DEST LINK	[symbolischen] Verweis („Link“) LINK –> DEST erzeugen
lpq	Drucke Warteschlangen-Status
lpr [–P QUEUE] FILE	Drucke FILE auf Drucker-Queue queue
ls [-la] [SPEC]	Aktuellen Verzeichnis-Inhalt als Liste von Dateinamen ausgeben. Übergebene Infos SPEC (Dateiname oder Verzeichnis) werden verwendet
man PROG	Beschreibung des Programms PROG
mkdir DIR	Verzeichnis erzeugen
more FILE	Textdatei FILE seitenweise anzeigen

mount –t TYPE DEVICE DIR	„Einhängen“ eines anderen Dateisystems vom Typ TYPE, das sich auf dem Gerät DEVICE befindet, in das Verzeichnis DIR
mv SOURCE DEST	Datei SOURCE in DEST umbenennen oder in Verzeichnis DEST verschieben *(wenn DEST ein Verzeichnis ist)*
passwd	Ändert das Passwort des aktuellen Benutzers
PROG	Ausführbares Programm PROG starten (wird in den in $PATH angegebenen Verzeichnissen gesucht)
PROG&	Programm PROG direkt als Hintergrundprozess starten (ohne Benutzereingaben)
ps [-ef]	Prozess-Informationen anzeigen
pstree [-ch]	Prozess-Informationen als Baumstruktur anzeigen (Eltern/Kinder)
pwd	Name des aktuellen Verzeichnisses ausgeben
rm [-i] FILE	Datei FILE löschen
rmdir DIR	Verzeichnis DIR löschen
sleep SEC	Hält die aktuelle Shell-Ausführung um SEC Sekunden an
sort [FILE] ..	Die Zeilen aller Textdateien sortiert ausgeben
su	Eine neue Shell im Superuser-Modus starten
sudo COMMAND	Nur den aktuellen Befehl COMMAND als Superuser ausführen
time PROG	Programm PROG starten und verbrauchte CPU-Zeit ausgeben
top [-d SECS]	Informationen über alle Prozesse ausgeben und nach SECS Sekunden aktualisieren (Default: 3)
VAR=VALUE	Shell-Variable VAR den Wert VALUE zuweisen
who	Aktuelle Benutzer dieses Systems anzeigen
> FILE	Standardausgabe (stdout) auf file umlenken, file ggf. neu erzeugen oder überschreiben
>> FILE	Standardausgabe (stdout) auf file umlenken, file ggf. neu erzeugen oder Ausgaben an file anhängen
$VAR	Die Zeichenkette $VAR durch den aktuellen Wert der Variablen VAR ersetzen
$1 $2 $3 …	Zeichenkette $1, $2, $3, .. durch jeweils 1., 2., 3. .. Parameter der Befehlszeile ersetzen ($0: Programmname)
$#	Zeichenkette $# durch Anzahl der Parameter der Befehlszeile ersetzen (Dezimalzahl)
$?	Zeichenkette $? durch return value des zuletzt aufgerufenen Programms (Vordergrundprozesses) ersetzen
$(PROG)	Das Programm PROG starten und die Zeichenkette $(PROG) durch die Ausgaben des Programms ersetzen
~	Wird durch den Inhalt von $HOME ersetzt
.	Zeiger auf das aktuelle Verzeichnis
..	Zeiger auf das direkt übergeordnete Verzeichnis
*	Metazeichen: Platzhalter für beliebig viele Zeichen
PROG1 | PROG2	Befehlsverkettung (Pipe): Ausgabe PROG1 = Eingabe PROG2