

INFO-Blatt - einige UNIX/LINUX Kommandos (in Kurzform)

1. Anfang und Ende einer Sitzung

Nach der Anwahl eines Rechners meldet sich automatisch der login-Prozess und fragt nach dem Benutzernamen (login:) und dem Passwort (password:). Wenn das Kennwort für den gegebenen Benutzer-Login korrekt ist, meldet sich der Befehlsinterpreter, auch Shell genannt, sofort mit dem Eingabeprompt. Stimmen die eingegebenen Daten nicht, so wird eine Fehlermeldung ausgegeben.

Das Passwort erscheint bei der Eingabe nicht auf dem Bildschirm. Das Passwort muss mindestens 6 Zeichen lang sein und mindestens 1 Sonderzeichen enthalten, empfehlenswert ist die Verwendung von Groß- und Kleinbuchstaben sowie Ziffern und Sonderzeichen.

Standardmäßig wird als Eingabeprompt ein \$ - Zeichen oder `/home/student>` verwendet, wobei `/home/student` das HOME-Verzeichnis (Hauptverzeichnis) für den Benutzer `student` ist.

Eine Sitzung beendet man entweder mit **logout** oder **exit**.

Befehle eingeben

Mit einem Befehl wird dem Betriebssystem mitgeteilt, welche Funktionen es ausführen soll. Die eingegebene Befehle werden von dem Befehlsinterpreter gelesen und anschließend ausgeführt. Jeder Befehl verfügt über eine eigene Syntax (*Befehl -Optionen Parameter*), in der die erforderlichen und wahlfreien Optionen, Dateien und Parameter festgelegt sind. Eine Option besteht aus einem Minuszeichen, gefolgt von den Einzelbuchstaben aller gewünschter Optionen ohne weitere Zwischenräume. Optionen sind durch Leerzeichen vom übrigen Befehl getrennt. Einige Programme, z.B. Compiler, verlangen jedoch Zwischenräume getrennte Angabe aller Optionen jeweils mit einem eigenen Minuszeichen.

Beispiel: `cc -s -o p p.c -lm`

Hinweise:

- Durch Verwendung eines Semikolons (;) können zwei Befehle in derselben Befehlszeile eingegeben werden.
- Bei Befehlen ist die Groß-/Kleinschreibung zu beachten.
- Ein sehr langer Befehl kann über mehrere Zeilen eingegeben werden, indem am Ende der Zeile ein umgekehrter Schrägstrich (Backslash) eingegeben und die Return-Taste gedrückt wird.
- Mit **Strg-C** werden die meisten Programme abgebrochen.

2. Hilfe

Jedes UNIX-Kommando kann im online-Handbuch *man* (für manual page, Handbuchseite) nachgeschlagen werden. Idealerweise liegt für jedes Kommando, jeden Aufruf der Programmierschnittstelle (Systemfunktionen, C-Bibliothek) und jede wichtige Konfigurationsdatei eine eigene man page vor.

Die Kommandos des Manpage-Systems:

Kommandos	Beschreibung
man	Handbuchseiten (MANual page). Ruft die Manpage (liefert eine ausführliche Beschreibung) für ein bestimmtes Programm, eine bestimmte Funktion, etc. auf.
whatis	Gibt eine einzeilige Kurzbeschreibung des Programms.
help	Einfache Hilfestellung für Anfänger.
apropos	Sucht in den Kurzbeschreibungen nach einer Zeichenfolge und gibt die passenden Einträge samt Kurzbeschreibung zurück. Nützlich wenn man sich z.B. an ein Kommando nicht mehr erinnern kann oder wenn man das Kommando für eine bestimmte Operation nicht kennt. Anstelle von <i>apropos</i> kann man auch <i>man -k</i> verwenden.

Beispielsweise gibt : `apropos mkdir` alle Einträge, in denen das Wort "mkdir" vorkommt.

Es gibt aber auch einen reinen info-Browser: `info (GNU) - Handbuch zu einem Kommando`.

3. Befehle oder Kommandos

3.1. Verzeichnisverwaltung

Dateien sind unter Unix (wie eigentlich auf allen modernen Betriebssystemen) in Verzeichnissen angeordnet. Das Unix-Dateisystem ist hierarchisch strukturiert und hat, grafisch dargestellt, die Form eines umgedrehten Baumes. Unter Unix gibt es nur einen einzigen Verzeichnisbaum, der Ausgangspunkt des Baumes ist die Wurzel (root), die mit / (slash), dargestellt wird und auch root-Verzeichnis genannt wird. Den Weg durch die Verzeichnisse zu der gewünschten Datei nennt man Zugriffspfad. Zu jeder Zeit des Dialogs verfügt man über ein aktuelles Verzeichnis (directory), dessen Namen man mit *pwd* erfährt. Nach dem login ist dies das Hauptverzeichnis (Home-Verzeichnis) des Benutzers. Die Angabe des Zugriffspfades bezieht sich entweder auf das Verzeichnis, in dem man sich gerade befindet oder auf das root-Verzeichnis (entscheidend dafür ist das erste Zeichen der Pfadangabe, s. *cd*).

In dem Verzeichnis */bin* liegen die am häufigsten benutzten Dienstprogramme. Die seltener benutzten Dienstprogramme befinden sich unter */usr/bin*.

Zur Verzeichnisverwaltung stehen u.a. folgende Befehle zur Verfügung:

Kommando	Beispiele	Beschreibung
<i>pwd</i>	(Print Working Directory)	Anzeigen des Namens des aktuellen Directories . (. Aktuelles Verzeichnis; .. Oberverzeichnis)
<i>cd</i>	(Change Directory)	Wechseln in ein anderes Unterverzeichnis (Subdirectory).
	<i>cd ..</i>	Wechseln zum jeweils nächsthöheren Directory.
	<i>cd prog/d1</i>	Wechseln zum (relativen) Subdirectory prog/d1.
	<i>cd /usr/bin</i>	Wechseln zum (absoluten) Directory /usr/bin.
<i>mkdir</i>	(Make DIRectory)	Erzeugen ein neues Verzeichnis.
	<i>mkdir Test1</i>	Es wird das neue Verzeichnis <i>Test1</i> angelegt.
<i>rmdir</i>	(ReMove DIRectory)	Entfernen (löschen) eines (<u>leeres</u>) Verzeichnis.
	<i>rmdir Test1</i>	Es wird das Verzeichnis <i>Test1</i> gelöscht.
<i>rm</i> (ReMove)	<i>rm -r subdir</i>	Entfernen des Unterdirectories <i>subdir</i> und aller seiner weiteren Unterdirectories (rekursiv).
<i>ls</i>	(LiSt)	Auflisten aller Dateien im aktuellen Directory - zusammen mit Punkt-Dateien.
	<i>ls -al</i>	Anzeigen aller Dateien (inklusive dot-Dateien) in Langform, d.h. mit zusätzlichen Dateiattributen.

3.2 Datei Kommandos

Zum Umgang mit Dateien stehen u.a. folgende Befehle zur Verfügung:

Kommando	Beispiele	Beschreibung
<i>ls</i> (LiSt)	<i>ls -a</i> <i>ls -g</i>	Auflisten aller Dateien im aktuellen Directory - außer dot-Dateien. Zeigt die Gruppenzugehörigkeit der Datei an.
<i>cat</i> (conCATenate)	<i>cat text</i> <i>cat t1 t2 >t</i>	Durchlaufende Anzeige der kompletten Datei <i>text</i> auf dem Bildschirm. Aneinanderhängen der Dateien <i>t1</i> und <i>t2</i> und ablegen in Datei <i>t</i>
<i>cp</i> (CoPy)	<i>cp sfile dfile</i>	Datei kopieren. Kopieren der Datei <i>sfile</i> (source file) zur Datei <i>dfile</i> .
<i>mv</i> (MoVe)	<i>mv ofile nfile</i>	Eine Datei verschieben oder umbenennen. Umbenennen der Datei <i>ofile</i> in <i>nfile</i> ; die Dateinamen können Pfadnamen sein.
<i>rm</i> (ReMove)	<i>rm file</i>	Löschen einer Datei. Entfernen der Datei <i>file</i> .
	<i>rm -i *.c *.h</i> <i>rm *.ch</i>	Entfernen aller Dateien mit Suffix <i>.c</i> und <i>.h</i> im Namen – mit Bestätigungsrückfrage für jede Datei (<i>i</i> Option: interactive mode).

Kommando	Beispiele	Beschreibung
Filter Programme		
more	<i>more file</i>	Die Datei <i>file</i> seitenweise anzeigen (mit der Leertaste weiterblättern, q=Ende).
less	<i>less file</i>	
head	<i>head -25 file</i>	Die ersten 25 Zeilen der Datei <i>file</i> anzeigen.
tail	<i>tail -5 file</i>	Die letzten 5 Zeilen der Datei <i>file</i> anzeigen.
sort	<i>sort file</i>	Sortieren aller Zeilen der Datei <i>file</i> in alphanumerisch aufsteigender Ordnung.
find	<i>find ./ -name stud -print</i>	Suchen und Anzeigen (print) einer Datei mit Namen <i>stud</i> vom aktuellen Directory an in allen Unterdirectories.
grep	<i>grep 'etwas' file</i> <i>grep -H main *.c</i>	Anzeigen aller Zeilen in der Datei <i>file</i> , die die Zeichenkette <i>etwas</i> enthalten. Anzeigen aller Dateien mit Suffix <i>.c</i> , die die Zeichenkette <i>main</i> enthalten.
wc (Word Count)	<i>wc file</i>	Auszählen aller Zeichen, Wörter und Zeilen in der Datei <i>file</i> .
touch	<i>touch file</i>	Letzte Dateiänderung auf akt. Datum setzen (Zugriffsdatum ändern).
	<i>cat file more</i>	Anzeige der Datei <i>file</i> seitenweise (eine Bildschirmseite) auf dem Bildschirm.
Pipes:	<i>cat file grep „stud“ wc -l</i>	Auszählen aller Zeilen in der Datei <i>file</i> , die die Zeichenkette <i>stud</i> enthalten.
>	<i>>file</i>	Umleitung der Ausgabe in Datei <i>file</i> .
<	<i><file</i>	Umleitung der Eingabe in der Datei <i>file</i> .
	<i>2>file</i>	Standardfehler nach <i>file</i> umleiten.
	<i>2>&1</i>	Standardfehler auf den Deskriptor 1 (stdout) umleiten.
	<i>>>file</i>	Standardausgabe nach <i>file</i> umleiten. Wenn die Datei existiert wird sie ergänzt und nicht überschrieben.
Dateien vergleichen		
cmp (CoMpare)	<i>cmp file1 file2</i>	Anzeigen der ersten Stelle in der Datei <i>file1</i> , an der ein Unterschied zur Datei <i>file2</i> vorkommt.
diff (Difference)	<i>diff file1 file2</i>	Anzeigen aller Unterschiede zwischen den Dateien <i>file1</i> und <i>file2</i> .
Archivierung		
tar (Tape Archive)	<i>tar -cvf Arh.tar *</i> <i>tar -xf Arh.tar</i>	Dateien können in einem Archiv abgelegt werden. Packen aller Dateien * in ein Archiv <i>Arh.tar</i> . Auspacken.
gzip	<i>gzip Archiv.tar</i>	(GNU) Kompressionstool.
Dateisystem		
du	(Disk Usage)	Anzeige des benötigten Speicherplatzes von Dateien und Verzeichnissen
df	(Disk Free)	Anzeige des freien Speicherplatzes auf Datenträgern
mount /umount	<i>mount /mnt /dev/hda2</i>	Dateisystem einhängen in den/aushängen aus dem Verzeichnisbaum
fsck	(FileSystem Check)	Dateisystem überprüfen.
mkfs	(MaKe FileSystem)	Dateisystem erstellen
Compiler	<i>cc -o exe file.c</i>	C Programme kompilieren.

3.3 Benutzer- und Rechteverwaltung

Unix ist von Anfang an ein Multi-User-Betriebssystem. Das bedeutet, dass verschiedene Benutzer am Rechner arbeiten können und voneinander abgeschottet werden: man kann anderen Benutzern erlauben oder verbieten, auf bestimmte Dateien zuzugreifen. Zusätzlich kann man Benutzer auch Gruppen zuordnen, denen kollektiv bestimmte Zugriffsrechte gewährt werden können. Eine besondere Rolle spielt der Benutzer root (Systemadministrator), der als einziger Benutzer vollen Zugriff auf das System hat.

<i>Kommandos</i>	<i>Beispiele</i>	<i>Beschreibung</i>
who	who	Zeigt die aktive Benutzer an.
su (Substitute User)	su - peter	Benutzer wechseln.
passwd		Ändern des eigenen Passwortes.

<i>Kommandos</i>	<i>Beispiele</i>	<i>Beschreibung</i>
chmod (CHange MODe)	<i>chmod a+r stud</i>	Einräumen (+) des Leserechts (read) an alle Benutzer (all) für die Datei oder das Directory <i>stud</i> .
	<i>chmod u-w,g-r stud</i>	Zurücknehmen (-) des Schreibrechts (write) für den Benutzer selbst (user) und des Leserechts für andere Gruppen-Mitglieder(group) für die Datei oder das Directory <i>stud</i> .
chown (Change OWNer)	<i>chown stud file</i>	Eigentümer und/oder Gruppe ändern
chgrp (Change GRouP)	<i>chgrp gruppe file</i>	Gruppenzugehörigkeit ändern
groups		Anzeige aller Gruppen, zu denen der Benutzer gehört.

3.4 Prozessmanagement (Steuerung von Prozessen)

<i>Kommandos</i>	<i>Beispiele</i>	<i>Beschreibung</i>
ps	ps -e	Anzeige (Status) der laufenden Prozesse.
kill	kill -9 7890	Einen Prozess beenden (Senden von Signalen an Prozesse).
killall		alle Prozesse des angegebenen Namens beenden
top (Table Of Processes)		interaktive Anzeige der laufenden Prozesse.
free		Anzeige des freien Speicherplatzes (flüchtiger Speicher).
nice renice (BSD)	nice [-incr] Kommando &	Priorität eines Prozesses ändern. Ausführen eines Hintergrundprozesses mit niedrigerer Priorität. <i>incr</i> gibt an, um wieviel die Priorität des Prozesses herabgesetzt werden soll (Voreinstellung: 10). Der Superuser kann Prozesse auch mit höherer Priorität laufen lassen, indem er ein negatives <i>incr</i> angibt.
pstree		Anzeige der laufenden Prozesse in Baumform.
&	ls -l &	Start eines Prozesses im Hintergrund. Alle Ausgaben von ls, auch die StandardFehlerausgaben, gelangen nach wie vor auf den Bildschirm, was u.U. die Arbeit im Vordergrund empfindlich stört (Umleitung nötig). Durch Ein-/Ausgabeumleitung kann der Hintergrundprozeß zum Schweigen gebracht werden: ls -l > DateienListe 2> /dev/null &
fg bg jobs		Befehle für die interne Job-Kontrolle der Bash.
nohup Kommando & (No Hang Up)		Einen Prozess nach dem Log-off weiterlaufen zu lassen. Automatische Umleitung in <i>nohup.out</i>

Der Start eines Prozesses im Hintergrund erfolgt durch anhängen eines **&** an die Befehlszeile. Zum Beispiel:

```
xterm & // startet ein X-Terminal im Hintergrund
```

Es wird die Processnummer (PID) des neu erzeugten Prozesses ausgegeben und die Shell meldet sich sofort wieder mit dem Eingabeprompt.

Ein laufendes Programm kann mit [Strg]+[Z] (^Z) in den Hintergrund verschoben werden; dort wird es zunächst angehalten ("suspend"). Das Kommando **jobs** gibt eine Liste aller derzeit im Hintergrund schlafenden Programme aus. Dabei wird für jedes Programm die interne Jobnummer angegeben. Es ist wichtig anzumerken, dass die vergebene Jobnummer in keinem Zusammenhang zu der angezeigten Prozessnummer steht. Mit **fg Jobnummer** holen Sie diese Task wieder in den Vordergrund. Soll ein angehaltenes Kommando dagegen im Hintergrund weiterlaufen, verwendet man den Befehl **bg Jobnummer**.

Zum Beispiel:

./p1	Programm p1 wird gestartet
<Ctrl+Z>	Prozess angehalten. System antwortet mit: <i>[1]+ Stopped .p1</i>
jobs -l	<i>[1]+ 5258 Angehalten .p1</i>
fg 1	Programm p1 (Jobnummer 1) läuft weiter in den Vordergrund
<Ctrl+Z>	
bg 1	<i>[1]- .p1 &</i> Programm p1 läuft weiter im Hintergrund

Beim Logoff werden auch alle Hintergrundprozesse gekillt, sofern nicht besondere Maßnahmen getroffen werden.

Wenn man einen Prozess mit **&** in den Hintergrund stellt und sich anschließend ausloggt, erhält dieser Prozess ein SIGHUP Signal. Dies führt, wenn der Programmierer es nicht explizit abfängt, normalerweise zum Abbruch des Programms. Um das Senden des Signals zu verhindern, gibt es den Befehl **nohup**, den man dem Befehl einfach voranstellt. Dadurch wird das Programm von SIGHUP nicht mehr belästigt und man kann sich problemlos ausloggen.

Bei der Verwendung des Befehls **nohup** ist abzumerken, dass die Ausgabe des Prozesses, die normalerweise auf das Terminal geht, in die Datei nohup.out (im zuletzt aktuellen Verzeichnis) umgeleitet wird. Falls der Prozess im aktuellen Pfad kein Schreibrecht hat, wird die Datei im Home-Directory abgelegt.

Prozess beenden

```
kill [-Signalnummer] [PIDs]
```

Mit diesem Kommando kann der Benutzer eigene Prozesse (in *PIDs* angegeben) beenden (auch im Vordergrund laufende Prozesse - von einem anderen Terminal aus, oder aus einem anderen X-Fenster heraus). Dem Prozess wird das Signal mit der beim **kill**-Aufruf angegebenen *Signalnummer* gesendet. Fängt der Prozess das Signal nicht ab, wird er terminiert.

Neben anderen können für **kill** folgende Signalnummern verwendet werden:

<i>Signal</i>	<i>Nr.</i>	<i>Beschreibung</i>
SIGHUP	1	Hangup (POSIX). (beim Beenden der Verbindung zum Terminal oder Modem)
SIGINT	2	Interrupt (ANSI). Terminieren des Prozesses (^C-Taste am Terminal)
SIGQUIT	3	Quit (POSIX).
SIGKILL	9	Kill (POSIX). Kann nicht abgefangen werden - Beendet immer den empfangenden Prozess.
SIGTERM	15	Termination (ANSI). Terminate (Software-Terminate, Voreinstellung)
SIGCONT	18	Continue (POSIX). Weiterführen des Prozesses (falls gestoppt).
SIGSTOP	19	Stoppen des Prozesses (^Z-Taste am Terminal)

Die Datei */usr/include/signal.h* (Linux: */usr/include/bits/signum.h*) enthält eine Liste aller Signale. Später wird das Kommando **trap** besprochen, mit dem man innerhalb von Shell-Skripts gezielt auf einzelne Signale reagieren kann.

4. Shell-Programmierung (Skript)

Ein bash-Skript ist quasi eine Batch-Datei unter DOS, aber leistungsfähiger. Die Hauptaufgabe der Shell besteht darin, Programme zu finden und auszuführen. Gefunden werden Programme, indem in allen Verzeichnissen, die in der Variablen *\$PATH* (s. **echo \$PATH**) stehen, nach ihnen gesucht wird. Ist ein Programm gefunden, so erstellt die bash eine Kopie von sich und ruft diese Kopie auf, um das gewünschte Programm auszuführen. Am Ende des Programms beendet sich auch die Kopie. Das Original hat solange gewartet, sofern man nichts anderes gesagt hat. Dieses Modell der Programmausführung ist typisch für Unix-Systeme.

Kommentare

Das Zeichen # bewirkt, dass der Rest der Zeile als Kommentar behandelt wird. Kommentare erhöhen die Lesbarkeit eines Programms und sollten daher reichlich verwendet werden.

Exit-Status

Der Exit-Status einer Befehlsfolge oder eines Skripts ist der Wert, den das letzte ausgeführte Programm liefert (siehe **echo \$?**). Zur Steuerung von bedingten Anweisungen, Schleifen, usw. dient der Exit-Status von Programmen. Will man in einem Skript selbst den Exit-Status setzen, so kann man dies mit **exit** wert tun (s. **exit**). Zwei Programme, die immer einen definierten Exit-Status erzeugen, sind **true** (immer 0) und **false** (immer 1).

Shell-Parameter

Die Shell kennt zwei Arten von Parametern: Positionsparameter und Shell-Variablen

Positionsparameter

Ihr Name wird als Ziffer 0,1,2 ,...,9 angegeben. Positionsparameter stellen die an ein Shell-Skript übergebenen Argumente zur Verfügung. Dem Parameter 0 wird der Name des aufgerufenen Shell-Skripts zugewiesen. Auf die Werte der einzelnen Parameternamen kann durch Voranstellen des **\$**-Zeichens zugegriffen werden. Mit dem Kommando **shift** können die Werte der Positionsparameter (nach links) verschoben werden.

Beispiele:

```
echo Der Skriptname ist $0
echo Das erste Argument ist $1
echo Das zweite Argument ist $2
```

Shell-Variablen

Neben den frei wählbaren Variablen bietet die Shell auch eine Reihe von Variablen an, deren Namen von ihr bereits fest vorgegeben sind. Bei diesen vordefinierten Variablen ist dann noch zu unterscheiden zwischen: vom Benutzer veränderbaren Shell-Variablen und Variablen, die ständig von der Shell automatisch gesetzt werden (auch automatische Variablen genannt).

Die **bash** kennt eine Reihe Variablen, die das Verwalten der **bash** parametrisieren. Der Befehl **set** gibt darüber Auskunft.

Durch Voranstellen von **\$** vor einem Parameternamen wird der Wert angesprochen, der unter diesem Parameternamen gespeichert ist.

Beispiele:

Tier = Hund; echo \$Tier # Benutzer Variable
echo \$PATH /usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/java/jre/bin
echo \$HOME /home/student
echo \$PS1 # Der Prompt \u@\h:\w>
echo \$PATH # Wie bash-Variablen verändert werden können, zeigt dieses Beispiel: /usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/opt/kde/bin:/usr/openwin/bin: PATH=/home/student/bin:\$PATH echo \$PATH /home/student/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/opt/kde/bin:/usr/openwin/bin:.

Die folgenden automatischen Variablen werden ständig neu von der Shell gesetzt:

Automatische Variablen	Bedeutung
#	Anzahl der gesetzten Positionsparameter.
@	Alle Positionsparameter als einzelne Strings: \$@ entspricht "\$1" "\$2" ...
*	Alle Positionsparameter als ein String: "\$*" entspricht "\$1 \$2 ..."
\$	Prozessnummer der aktuellen Shell.
!	Prozessnummer des zuletzt im Hintergrund gestarteten Kommandos.
?	Exit-Status des zuletzt im Vordergrund ausgeführten Kommandos.
-	Optionen, welche beim Aufruf der Shell angegeben oder mit dem <i>set</i> Kommando eingeschaltet wurden.

Kommando-Strukturen

IF b; THEN ; ELSE ; FI IF b THEN ; [ELIF b THEN -;] [ELSE ;] FI	Das Skript überprüft, ob der erste Argument ein Verzeichnis ist.
Die <u>bedingte Anweisung</u> : if bedingung then kommandos1 else kommandos2 fi führt die Anweisung(en) bedingung aus. Falls der <i>Exit</i> -Status <i>true</i> ist, werden die Befehle nach then , sonst die nach else ausgeführt. Der <i>else-Teil</i> kann auch fehlen. Es gibt auch eine erweiterte Form.	if [! -d "\$1"] then # Wenn der erste Parameter kein Verzeichnis ist, dann echo "\$0: \$1 ist kein Directory" >&2 exit 1 fi echo -n "\$1" Achtung! Die Syntax der If-Else Anweisung muss genauso geschrieben werden (die Leerzeichen sind EXTREM wichtig !)

WHILE b; DO -; DONE	Das Skript zeigt alle Argumente an.
Die Kommandos in einer <u>while</u> -Schleife while bedingung do kommandos done werden ausgeführt, solange die Bedingung true ist.	while ["\$#" -ge 1] do echo \$1 shift done

UNTIL b; DO -; DONE	Das Skript führt den sleep-Befehl solange aus, bis sich der als Argument angegebene Benutzer eingeloggt hat.
Im Gegensatz zur while-Schleife werden in der until- Schleife until bedingung do kommandos done die Befehle ausgeführt, solange die Bedingung false ist.	until who grep \$1 # Exitstatus von grep do sleep 10 done echo "\$1 ist jetzt da"

FOR v IN liste DO -; DONE	Wenn der erste Parameter ein Verzeichnis ist, erzeugt das Skript eine Liste des Inhaltes des Verzeichnisses.
Eine for -Schleife erhält eine Liste von Werten und führt die Kommandos mit jedem dieser Werte in der Laufvariablen aus. for var in liste do kommandos done	arg= for i in \$1/* do arg="\$arg \$i" # einfügen done echo \$arg
	Backup Dateien erzeugen: for i in *.doc; do cp \$i \$i.bak; done

Schleifen können vorzeitig abgebrochen werden:

break [n] bricht eine (bzw. die nächsten n) **while**-, **for**- oder **until**-Schleife ab.

continue [n] unterbricht den aktuellen Schleifendurchlauf und geht an den Anfang der **while**-, **for**- oder **until**-Schleife zurück.

CASE v IN ... ESAC	Das Skript prüft die Optionen -l und -s
Die Variable einer <u>case</u> -Anweisung case var in muster1) kommandos1 ;; muster2) kommandos2 ;; ... esac wird der Reihe nach mit den Mustern verglichen und nur die zum ersten passenden Muster gehörenden Befehle werden ausgeführt. Die Muster haben die gleiche Syntax wie die Dateinamen-Expansion. Alternativen können durch angegeben werden. Das Muster * trifft immer zu und dient am Ende der Anweisung dazu, Befehle auszuführen, wenn keins der anderen Muster passte.	while getopt ls: opt do case \$opt in l) Lflag=1;; s) OPT=\$OPTARG;; \?) usage exit 1;; esac done

Benutzereingaben

Der Befehl:

```
read var1 [var2 ...]
```

Beispiel:

```
echo "Wie heisst Du ?"  
read Name  
echo "Hallo $Name"
```

liest eine Zeile von der Standard-Eingabe und weist die einzelnen Worte den Variablen zu. Wenn mehr Worte als Variablen vorhanden sind, erhält die letzte Variable den Rest der Zeile; wenn die Eingabe zu kurz ist, bleiben die letzten Variablen leer.

Quoting, Maskieren

Bestimmte Zeichen haben für die bash eine besondere Bedeutung. Will man diese Zeichen in ihrer literalen Bedeutung nutzen, so muss man ihre spezielle Wirkung maskieren (quoting). Die bash kennt drei Formen des Maskierens:

" ... "	Mit Ausnahme des \$ (Variablen- und Befehlssubstitution) und des \ sind alle Zeichen maskiert.
' ... '	Alle Zeichen sind maskiert.
\	Das folgende Zeichen wird maskiert. Kann auch innerhalb doppelten Anführungszeichen verwendet werden.

Tipp: Man kann viel lernen wenn man sich fertige Skripte ansieht.

Platzhalterzeichen in Dateinamen

Dateinamen können Platzhalterzeichen enthalten und bilden ein Dateinamen-Suchmuster. Dieses Muster wird vor der Befehlsausführung mit den vorhandenen Dateien verglichen (globbing). Die auf das Muster passenden Dateinamen werden zu Befehlsargumenten. Diesen Vorgang nennt man auch Parameter-Expansion.

<i>Platzhalter</i>	<i>Bedeutung</i>
*	Passt auf jede Zeichenfolge. Ein *, der als erstes Zeichen eines Dateisuchmusters steht, passt nicht auf Dateinamen, die mit einem Punkt beginnen.
?	Passt auf genau ein beliebiges Zeichen.
[xyz]	Passt auf genau ein Zeichen aus der Menge der in eckigen Klammern eingeschlossenen Zeichen.
[! xyz]	Passt auf jedes Zeichen, das nicht in der Menge der in eckigen Klammern eingeschlossenen Zeichen ist

Editoren (vi, joe)

Überall ist der Editor <vi> installiert. Er ist aber sehr gewöhnungsbedürftig. Daher sollten Sie joe nehmen, der mit den alten Wordstar-Kommandos schnell zu bedienen ist.

joe beispiel.txt startet den Editor mit der Datei *beispiel.txt*, die entweder geöffnet oder neu angelegt wird, wenn die Schreibrechte für das Verzeichnis stimmen. Mit <STRG> + <K> + <H> blenden Sie die Hilfe ein bzw. aus. Die Texteingabe muss nicht weiter erläutert werden.

Wenn Sie mit dem Speichern beenden wollen geben Sie <STRG> + <K> + <X>, wenn sie den Editor ohne Speichern verlassen wollen geben Sie <STRG> + <C> ein. Im letzten Fall müssen Sie noch mit <Y> bestätigen, dass Sie die Änderungen nicht speichern wollen.

VI-Editor Tutorial: <http://tutorials.beginners.co.uk/read/category/11/id/269>

Folgendes ist eine unvollständige Zusammenstellung wichtiger Unix-Kommandos.

<i>Unix-Kommandos (Alphabetisch geordnet)</i>	<i>Kurze Beschreibung</i>
ar d m p q r t x as [-o objfile] file at time [date] [script] atq [-c] [-n] user atrm [-fi] [-] [job-no] [user] awk basename string [suffix] bc [-c] [-l] [file] cal [month] year cat cb [-js] [-l leng] file	Archiv Verwaltung von Bibliotheken (lib) Assembler Kommandoausfuehrung zu angegebener Zeit Anzeige der "at"-Warteschlange Jobs aus "at"-Warteschlange entfernen Prozessierung von Zeichenmustern in Dateien Extrahiert Dateinamen aus Pfadnamen Interpreter für arithmetische Sprache Ausgabe eines Kalenders Verketteten und Ausgabe von Dateien Übersichtliche Ausgabe von C-Programmen
cc [option ...] file cd directory cflow file ... chgrp group file dir chmod mode file chown user file dir chroot directory command cmp [-l] [-s] file1 file2 comm [-123] file1 file2 compress file cp [-ip] file1 [file2 dir]	C-Compiler aufrufen (compile, link) Verändern des aktuellen Directories Übersicht über externe Referenzen in Quelldateien Gruppenzugehörigkeit von Dateien ändern Schutzinformationen (a ugo - += rwx st) Eigentümer ändern Ausgangspunkt für abs. Pfadnamen Vergleich zweier Dateien Suche gemeinsamer Zeilen in 2 sortierten Dateien Lempel-Ziv Datenkompression (.Z) Dateien kopieren
date dc [file] dd if=infile of=outfile [conv=...] df [filesystem] diff [-b] [-e -h] file1 file2 diff3 [-ex3] file1 file2 file3 dirname [pathname] du [-s] [-a] [-r] [file dir] dump echo [argument]	Anzeige/Veränderung von Systemdatum und -zeit "desk calculator" (UPN, Postprozessor für bc) Kopieren und Konvertieren von Dateien Anzeige des freien Speicherplatzes (auf Disk) Vergleich von Text-Dateien Vergleich von drei Text-Dateien Entfernung der letzten Komponente des Pfadnamens "disk usage" belegte Datenblöcke Backup oder Ausgabe von Symboltab. Ausgabe einer Meldung
ed [-] egrep [...] pattern [file] env [-] [name=value] [command] expr expression fdformat [-deflv] [-b label] [device] fgrep [...] pattern [file] file [-f ffile] file ... find pathname condition action finger [name] fschk	zeilenorientierter Editor "grep" mit vollst. Regular Expressions Anzeige bzw. Änderung von Umgebungsvariablen Auswertung der Argumente Formatierung von Disketten "grep" mit reinen Zeichenketten Bestimmung des Datei-Typs Suchen von Dateien und Kommando-Ausführung Information über Benutzer "file system consistency check" prüft Dateisysteme
ftp [...] [hostname] gcc gdb getopt grep [-cilnv] pattern [file] groups [username] gzip head [-n] file	Datei-Übertragung via TCP/IP GNU C Compiler GNU Debugger Parser für Kommandozeilen-Optionen Suche von Zeichenfolgen in Dateien Gruppenzugehörigkeit GNU Kompressionstool Ausgabe der ersten n Zeilen einer Datei
hostid hostname id joe join [...] file1 file2 kill [-signal -l] PID ld [...] file.o ... lex [-ntv] [file] line lint [...] file.c ... ln [-fs] filename linkname	Maschinen-Identifikation IP-Name der Maschine Ausgabe der Benutzer- und Gruppennummer Editor Verknüpfen von Zeilen von sortierten Dateien Senden von Signalen an Prozesse Linken von Objektmodulen Erzeugung von Programmen für lexikal. Analysen Lesen einer Zeile von <i>stdin</i> , Ausgabe nach <i>stdout</i> Prüfen von C-Programmen (Portabilität etc.) Aliasname bzw. symbolischer Link

<i>Unix-Kommandos (Alphabetisch geordnet)</i>	<i>Kurze Beschreibung</i>
login [user] logname look [-df] string file lp [...] file lpq lpr lprm lpstat lptest ls [-CFRabcdgilrstu] file dir	Anmelden als Benutzer Ausgabe des Login-Namens Durchsuchen einer sortierten Datei Ausdrucken von Dateien Ausgabe der Drucker-Warteschlange Druckauftrag abschicken Druckauftrag entfernen Informationen über lp-Spoolsystem Erzeugt Testmuster für Drucker Anzeige von Datei-Informationen (directory)
mail make [-f makefile] [macro=value] man [section] name mesg [y/n] mkdir dir mkfs special size mknod file [b c p] major minor more [file] mount [...] [filesystem directory] mv file dir newname	Absenden und Empfangen elektronischer Post Zieldatei erstellen On-Line-Manual Zulassen oder Abblocken von "write"-Meldungen Erstellen von Directories Erzeugen eines (leeren) Dateisystems (su) Erzeugt eine Gerätedatei Seitenweises Anzeigen von Dateien Ankoppeln eines Dateisystems (su) Umbenennen oder Verschieben von Dateien
netstat newfs [...] raw-special-device newgrp [-] group nice [priority] command nl [...] [file] nm [...] [file] nohup command nroff [-man ...] file od [...] [file] passwd [user] paste [-s] [-dliste] file1 file2 ...	Netzwerk-Status anzeigen Erzeugen eines neuen Dateisystems (su) Ändern der aktuellen Gruppenzugehörigkeit Scheduling-Priorität ändern Zeilen-Numerierung Ausgabe von Symboltabellen aus Objektmodulen nicht unterbrechbare Ausführung eines Kommandos Textformatierung (von Manuals) "Octal Dump" Dateiausgabe in wählbarem Format Passwort ändern horiz. Aneinanderfügen der Zeilen
pr [...] [file] ps pstree pwd rcp [-r] from to rlogin host [-l user] rm [-fir] file ... rmdir dir ...	formatierte Ausgabe Informationen über aktive Prozesse Anzeige der laufenden Prozesse in Baumform. Ausgabe des vollst. Pfadnamens des akt. Directories "remote copy" zwischen UNIX-Rechnern Login auf einem fernen UNIX-Rechner Löschen von Dateien leere Directories löschen
rpcgen rsh host [-l user] [-n] command rwho [-a] sed sh shutdown sleep seconds sort [-dfur...] [file] spell [-b] [-v] [-d hlist] [...] [file] split [-n] [infile [outfile]] strip [-x] [-r] file	RPC-Protokoll-Compiler Kommando-Ausführung auf anderem Rechner Anzeige der eingeloggten Benutzer im LAN "Stream Editor" Nicht-interaktiver Editor Kommando-Interpreter (Bourne-Shell) Herunterfahren des Systems Warten, bis bestimmte Zeit abgelaufen ist Sortieren von Dateien Rechtschreibung Prüfung Aufteilen einer Datei in kleinere Entfernen der Symboltabelle aus Lademodulen

<i>Unix-Kommandos (Alphabetisch geordnet)</i>	<i>Kurze Beschreibung</i>
stty su [-] [user] sync tabs [tabspec] [-Ttype] [+mn] tail [+/-n[bcl][fr]] [file] tar c r t u x[...][f file] tee [-a] file telnet [host [port]] test expression touch [-amc] file tput [-Tterm] name tr [-cds] [string1 [string2]] trace [-ct] [-o fname] command troff [...] [file] tty [-s]	Anzeige und Veränderung von Terminal-Parametern vorübergehende Änderung der Benutzer Schreibt veränderten Superblock auf Platte Setze Tabulator-Stops am Terminal Ausgabe des Endes einer Datei Sicherung und Rücksicherung von Dateien Schreibe auf Standard-Output und in Datei Login auf einem fernen Rechner via TCP/IP Test zur Ablaufsteuerung in Prozeduren Zugriffs- oder Modifikationsdatum ändern Ausgabe von Werten aus terminfo-Beschreibungen Zeichen-Ersetzung Spürt System-Aufrufe auf Textformatierung für Fotosatzmaschinen Ausgabe der Pfadbezeichnung des Terminals
umask umount filesystem dir uname [-a] [-snrvm] uncompress file[.Z] uptime uucp [-cdm] infile outfile uudecode [encoded-file] uuencode [source-file] file-label unzip vedit vi [-R] [-t tag] [-wn] [-r file] [+cmd] file w [-hls] [user] wait	Die Zugriffsberechtigungsmaske ändern. Abkoppeln eines Dateisystems Ausgabe von Informationen über Hardware und BS Lempel-Ziv Datendekompression Zeit seit reboot und Systemauslastung Unix-to-Unix-Copy Decodierung von ASCII nach Binär Codierung von Binär nach ASCII s. gzip s. vi Bildschirm-Editor "who" eingeloggte User und deren Aktivität Warten auf Beendigung von Hintergrundprozessen
wall wc [-clw] [file] what [-s] filename whatis name which command who write user [terminal] xget xsend yacc [-dv] grammar yppasswd ypwhich zcat file.z	Senden von Meldungen an alle Benutzer "Word Count" Zählen von Zeilen, Wörtern, Zeichen Anzeige der SCCS-Information Kurzinformation über ein Kommando Kommando lokalisieren (in Pfad oder Alias) Ausgabe der eingeloggten Benutzer Senden einer Mitteilung Lesen einer verschlüsselten Mail Abschicken einer verschlüsselten Mail "Yet Another Compiler-Compiler" Ändern des Netzwerk-Passworts Hostname des NIS-Servers Anzeige (cat) einer komprimierten Datei ("pack")