Übungen Betriebssysteme I
Wintersemester 2005

UNIVERSITÄT SIEGEN

Fachbereich 12,
Elektrotechnik und Informatik
Betriebssysteme / verteilte Systeme
Dipl.-Ing. Adrian Kacso

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 gennant, sofort mit dem Eingabeprompt. Stimmen die eingegebenen Daten nicht, so wird 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>* , 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 festlegen sind. Eine Option besteht aus einem Minuszeichen, gefolgt von den Einzelbuchstaben aller gewünschter Optionen ohne weitere Zwischenräume angegeben werden. 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:

a) Durch Verwendung eines Semikolons (;) können zwei Befehle in derselben Befehlszeile eingegeben werden.
b) Bei Befehlen ist die Groß-/Kleinschreibung zu beachten.
c) 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.
d) 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 *apropos* kann man auch *man -k* 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.
Folgendes ist eine unvollständige Zusammenstellung wichtiger Unix-Kommandos.

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 der Wurzel (root), die mit / (slash), dargestellt wird und auch root-Verzeichnis genannt wird. Der 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 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***
pwd	(Print Working Directory)	Anzeigen des Namens des aktuellen Directories . **(.** Aktuelles Verzeichnis; **..** Oberverzeichnis)
cd	Change Directory)	Wechseln in ein anderes Unterverzeichnis (Subdirectory).
	cd ..	Wechseln zum jeweils nächsthöheren Directory.
	cd prog/d1	Wechseln zum (relativen) Subdirectory prog/d1.
	cd /usr/bin	Wechseln zum (absoluten) Directory /usr/bin.
mkdir	*(Make DIRectory)*	Erzeugen ein neues Verzeichnis.
	mkdir Test1	Es wird das neue Verzeichnis *Test1* angelegt.
rmdir	(ReMove DIRectory)	Entfernen (löschen) ein (leeres) Verzeichnis.
	rmdir Test1	Es wird das Verzeichnis *Test1* gelöscht.
rm (ReMove)	*rm -r subdir*	Entfernen des Unterdirectories *subdir* und aller seiner weiteren Unterdirectories (rekursiv).
ls	(LiSt)	Auflisten aller Dateien im aktuellen Directory - außer dot-Dateien.
	ls -al	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***
ls (LiSt)	*ls -a*	Auflisten aller Dateien im aktuellen Directory - außer dot-Dateien.
	ls -g	Zeigt die Gruppenzugehörigkeit der Datei an.
cat (conCATenate)	*cat text*	Durchlaufende Anzeige der kompletten Datei *text* auf dem Bildschirm.
	cat t1 t2 >t	Aneinanderhängen der Dateien *t1* und *t2* und ablegen in Datei *t*
cp (CoPy)	*cp sfile dfile*	Datei kopieren. Kopieren der Datei *sfile* (source file) zur Datei *dfile*.
mv (MoVe)	*mv ofile nfile*	Eine Datei verschieben oder umbenennen. Umbenennen der Datei *ofile* in *nfile*; die Dateinamen können Pfadnamen sein.
rm (ReMove)	*rm file*	Löschen einer Datei. Entfernen der Datei *file*.
	rm -r subdir	Entfernen des Unterdirectories *subdir*.
	*rm -i *.c*	Entfernen aller Dateien mit Suffix .c und .h im Namen - mit Bestätigungsrückfrage für jede Datei (interactive mode).

Kommando	*Beispiele*	*Beschreibung*
Filter Programme		
more less	*more text* *less text*	Die Datei *text* seitenweise anzeigen (mit der Leertaste weiterblättern, q = Ende).
head	*head -25 text*	Die ersten 25 Zeilen der Datei *text* anzeigen.
tail	*tail -5 text*	Die letzten 5 Zeilen der Datei *text* anzeigen.
sort	*sort file*	Sortieren aller Zeilen der Datei *file* in alphanumerischaufsteigender Ordnung.
find	*find ./ -name stud -print*	Suchen und Anzeigen (print) einer Datei mit Namen *stud* vom aktuellen Directory an in allen Unterdirectories.
grep	*grep 'etwas' file* *grep 'main' *.c*	Anzeigen aller Zeilen in der Datei *file*, die die Zeichenkette *etwas* enthalten. Anzeigen aller Dateien mit Suffix .c, die die Zeichenkette *main* enthalten.
wc (Word Count)	*wc file*	Auszählen aller Zeichen, Wörter und Zeilen in der Datei *file*.
touch	*touch file*	Letzte Dateiänderung auf akt. Datum setzen (Zugriffsdatum ändern).
| Pipes:	cat file | more cat file | grep „stud“ | wc -l	Anzeige der Datei *text* seitenweise (eine Bildschirmseite) auf dem Bildschirm. Auszählen aller Zeilen in der Datei *file*, die die Zeichenkette *stud* enthalten.
>		Umlenkung der Ausgabe
<		Umlenkung der Eingabe
Dateien vergleichen		
cmp (CoMpare)	*cmp file1 file2*	Anzeigen der ersten Stelle in der Datei *file1*, an der ein Unterschied zur Datei *file2* vorkommt.
diff (Difference)	*diff file1 file2*	Anzeigen aller Unterschiede zwischen den Dateien eins und zwei.
Archivierung		
tar	*(Tape Archive)* *tar -cfv Arch *.c* *tar -xf Arch.tar*	Archivierungsprogramm. Dateien können in einem Archiv abgelegt. Komprimieren alle Dateien mit Suffix .c in einem Archiv. Unkomprimieren.
gzip	*gzip Archiv.tar*	(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	*mount /mnt / dev/hda2*	Dateisystem einhängen in den/aushängen aus dem Verzeichnisbaum
fsck	*(FileSystem CheckK)*	Dateisystem überprüfen.
mkfs	*(MaKe FileSystem)*	Dateisystem erstellen
Compiler	*cc -o exe file.c*	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 voreinander 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.

Kommandos	*Beispiele*	*Beschreibung*
who	who	Werden die aktive Benutzerkennungen angezeigt.
su (Substitute User)	su - peter	Benutzer wechseln.
passwd		Ändern des eigenen Passwortes. Regeln für Passwörter: mindestens 6 Zeichen, darunter mindestens 1 Sonderzeichen

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

3.4 Prozessmanagement (Steuerung von Prozessen)

Kommandos	*Beispiele*	*Beschreibung*
ps	ps	Anzeige (Status) der laufenden Prozesse.
kill		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. *incr* 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 *incr* angibt.
pstree		Anzeige der laufenden Prozesse in Baumform.
&	ls -l &	Start eines Prozesses im Hintergrund. Alle Ausgaben von ls, auch die Fehlerausgaben, 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 Logoff weiterlaufen zu lassen. Automatische Umleitung in *nohup.out*

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] 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: *[1]+ Stopped ./p1*
jobs -l	*[1]+ 5258 Angehalten ./p1*
fg 1	Programm p1 (Jobnummer 1) läuft weiter in den Vordergrund
<Ctrl+Z>	
bg 1	*[1]- ./p1 &* 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 abzusehen, dass die Ausgabe des Prozesses, die normalerweise auf das Terminal geht, wird in die Datei nohup.out (im zuletzt aktuellen Verzeichnis) umgeleitet. 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 Vordergund 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 Nummer gesendet (Voreinstellung: 15). Fängt der Prozess das Signal nicht ab, wird er terminiert.

Neben anderen können für kill folgende Signalnummern verwendet werden:

Signal	***Nr.***	***Beschreibung***
SIGKILL	0	Terminate (beim Beenden der shell)
SIGHUP	1	Hangup (beim Beenden der Verbindung zum Terminal oder Modem)
SIGINT	2	Interrupt (wie Ctrl-C-Taste am Terminal)
SIGQUIT	3	
SIGKILL	9	Kann nicht abgefangen werden - Beendet immer den empfangenden Prozess.
SIGTERM	15	Terminate (Software-Terminate, Voreinstellung)

Die Datei /usr/include/Signal.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. 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.

Beispiel:

```
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 /home/stud/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/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 *set* 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 bedingte Anweisung: **if** bedingung **then** kommandos1 **else** kommandos2 **fi** führt die Anweisung(en) bedingung aus. Falls der *Exit*-Status *true* ist, werden die Befehle nach **then**, sonst die nach **else** ausgeführt. Der *else- Teil* 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 **while** -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] 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 **case** -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 wird 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** getopts 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.

Platzhalter	*Bedeutung*
*	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

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

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

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