



Referenz:

bash

(Kommandozeile)

Sonderzeichen:

Whitespaces:	SPACE \t \n
Wildcards:	? * [...]
E/A-Umlenkung:	< > << >>
Kommandotrennung:	&& ; (...)
Shell-Variablen:	\$ \${...}
Befehlssubst:	`...` \$(...) \${(...)}
Entwertungszeichen:	"..." '! ' \

-> Eingabezeile

Shell interpretiert
Sonderzeichen

+ Befehlszeile

Sichtbar machen mit: `set -x`
Ausschalten mit: `set -`

Version V 7.02

© Menne, b.i.b. Paderborn

Linux-Grundlagen

Umgang mit der Kommandozeile

↑↓	Cursortasten: letzte Befehle anzeigen/bearbeiten
→	Tab: Angefangenen Befehl/Dateinamen ergänzen
^C	Abbruch von laufenden Programmen
AD	Textendezeichen für manuelle Texteingabe

Verzeichnis-Kommandos

ls	Verzeichnis auflisten lassen, Optionen:	
-l	langes Listing	-a auch versteckte Dateien
-F	Dateityp darstellen	-i Inodes anzeigen
-d	Directory ohne Inhalt	-R alles ab diesem Verzeichnis
cd verzeichnis	In ein Verzeichnis wechseln	
pwd	aktuelles Verzeichnis anzeigen (Print Working Directory)	
mkdir verzeichnisse	Verzeichnis erstellen	
-p	auch alle Zwischenverzeichnisse neu erstellen	
rmdir verzeichnis	Leeres Verzeichnis löschen	
rm -rf verzeichnis	Verzeichnis mit Inhalt sofort löschen	

Verzeichnissymbole:

~	Home-Verzeichnis (wie \$HOME, Bsp: <code>ls ~dozmen/Public</code>)
/	Root-Verzeichnis (<code>ls /</code>)
.	Name des aktuellen Verzeichnisses
..	Name des über dem aktuellen liegenden Verzeichn.

Verzeichnis/Dateinamenssymbole (Joker | Wildcards):

*	beliebig viel Zeichen.	Bsp: <code>cp * *</code>
?	genau ein Zeichen.	Bsp: <code>more test?.txt</code>
[a-z]	ein Zchn von a bis z.	Bsp: <code>vi [a-d]*.c</code>
[!a-z]	kein Zchn von a bis z.	Bsp: <code>ls [!a-zääü]*</code>

Datei-Kommandos

more dateien	Datei-Inhalte seitenweise ausgeben
cat dateien	Datei-Inhalte ausgeben
-v	Sonderzeichen sichtbar machen
rm dateien	Dateien/Verzeichnisbäume löschen
-R	rekursiv löschen (-r)
-f	force: ohne nachzufragen bei schreibgeschützten Dateien
cp dateien verzeichnis	cp datei datei
	Dateien in ein Verzeichnis kopieren oder Datei kopieren
-R	rekursiv kopieren
-i	vor Überschreiben nachfragen
mv dateien verzeichnis	mv datei datei
	Umbenennen/Verschieben von Dateien oder Verzeichnissen
ln -s dateipfad verzeichnis	symbolischen Link erstellen
ln von nach	Hardlink erstellen
touch dateien	Datei erstellen/aktualisieren
lp datei	Datei ausdrucken

Bedeutung von Zeichen aufheben

Sonderbedeutung von Zeichen wie: *, ?, &, ;, <, >, ... aufheben:	
\\$	Maskierung des nachfolgenden Zeichens \$
'...'	Maskierung der Sonderzeichen incl. \$ und \
"..."	Maskierung der meisten Sonderzeichen außer \$ und \

Ausgabe

echo	zeigt alle Parameter durch Leerzeichen getrennt an
echo -n	Zeilenvorschub unterdrücken
echo -e	bash: Sonderzeichen interpretieren:
\a	Ton
\b	backspace
\r	Anfang Zeile
\t	Tab
\n	Zeilenvorschub
\033	Oktaal-Codes (\0x1)
Bsp: echo -e	"Menne:\t 1234"
tabs 20	Tabulatoren der Eingabezeile setzen
printf	Formatierte Anzeige im C-Stil, Formatzeichen:
%20s	String rechtsbündig mit 20 Zchn (auch "%20.10s")
%-20s	String linksbündig mit 20 Zchn
%8d	Zahlen auf 8 Stellen
Bsp: printf	"%-10s %8.2f %3s \n" "Ergebnis:" 12,34566 "€"

Linux-Info's

man befehl	Hilfe-Manual (z.B. <code>man ls</code>)
(„leertaste“ für nächste Seite, „q“ für Ende, „h“ für Hilfe)	
-k	suchwort
man ksh col -bx	Alle Kommandos zum Suchwort zeigen
man ksh col -bx	Manual ohne Formatierungen erzeugen
date:	Ausgabe des aktuellen Datums mit Uhrzeit.Formatierungen:
date +%d.%m.%Y	deutsches Datumsformat (=%)
date +%j-ter Tag %A"	Jahrestag und Wochentag
date -d "2007-08-03" +%j	Jahrestag ermitteln
date -d "-45 days"	45 Tage vorher
Datumsdifferenz in Tagen:	
\$((\$(date -d '2018-05-24 0:00' +%s) - \$(date -d '0:00' +%s)) / 86400)	

Shell-Zeichen

space, tab	Optionentrenner (auch Zeilenvorschub!)
~	Pfad des HOME-Verzeichnisses (auch: "~dozmen")
~	Befehlstrennung
;	
(...)	Kombination von Linux-Befehlen: (<code>ls; who</code>) <code>more</code>
&&	Im Erfolgsfall: <code>grep men * && echo "ok"</code>
	Im Fehlerfall: <code>grep men * && echo "---"</code>

Interessante Linux-Befehle

clear	Bildschirm löschen
cal 1 2000	Kalender anzeigen (Deutsch: <code>cal -ym</code>)
who	Liste der eingeloggten User
vi -x test.txt	Datei verschlüsseln mit Passwort
quota -v	Anzeige des verbrauchten Speicherplatzes
du -sh ~/*	Speicherplatz der Verzeichnisse summieren
find ~ -user yyy -size +1000k -exec ll {} \;	Große Dateien suchen!
mail -s „Hallo“ men@bib.de < brief.txt	Email verschicken
mail -s „Hallo“ men@bib.de -a ich.jpg < brief.txt	Email mit Anhang
script	Bildschirmprotokoll erstellen
history	Die letzten Befehle anzeigen lassen
time command	Zeitmessung (Tipp: <code>export TIMEFORMAT=\"%R sec\"</code>)

Prozesse stoppen:

ps	Liste der Prozesse (auch: top)
-e	alle Linux-Benutzern (every) -l erweiterte Anzeige (long)
-f	erweitertes Listing (full)
kill -9 4711	löscht den Prozess 4711

Befehlsabkürzungen:

alias ll="ls -l"	Kommandoabkürzung vereinbaren
unalias name	Kommandoabkürzung löschen

Zeichenkodierung ändern:

recode pc < msdos.txt	Zeilenformat umcodieren
dos2unix programm.sh	Zeilenformat ändern
recode /cr-lf..latin1 <windows.txt	Windowsdatei umcodieren
recode latin-1..UTF-8 datei	Ins UTF-8-Format umwandeln

Bash-Konfiguration

export LANG=de_DE.UTF-8	Deutsche Sprach-Einstellung
set -o	Gesetzte Optionen anzeigen lassen
set +o histexpand	Option zurücksetzen (hier: <code>echo „Hallo!“</code>)
~/bashrc	Persönliche Datei mit Login-Kommandos
~/bash_logout	Persönliche Datei mit Logout-Kommandos
~/exrc	Datei mit vi-Konfigurierungskommandos
~/sh_history	Datei mit den letzten Befehlen

Zugriffsrechte

Bei Dateien:	Bei Verzeichnissen:
r lesbar	x betretbar (cd)
w bearbeitbar	r lesbar (ls, find)
x ausführbar	w dateien/verz erstellen/löschen
chmod benutzergruppe[+ - =]zugriffsrechte dateien	
(u=user, g=group, o=others, a=all) (+ hinzu, - weg, = absolut)	
Bsp.: <code>chmod 751 *.sh *.cmd</code> ; <code>chmod og+rx *.sh</code>	
(StickyBits: u+s, g+s, o+t oder z.B. <code>chmod 4744</code>)	
getfacl datei	Zugriffsrechtlisten anzeigen
setfacl -m dozmen:rw datei	Zugriffsrecht für Benutzer vergeben
setfacl -m g:doz:r datei	Zugriffsrecht für Gruppe vergeben
setfacl -x dozxxx datei	Benutzer aus ACL-Liste löschen

E/A-Umlenkung

Kanäle: 0 stdin, 1 stdout, 2 stderr	
< datei	stdin aus datei holen
1> datei	stdout in Datei
1>> datei	stdout an Datei anhängen
2> datei	stderr umlenken
2>&1 >datei	stdout und stderr zusammen umlenken (kurz: "&>")
2>/dev/null	stderr in den Mülleimer
prog1 prog2	stdout von prog1 an stdin von prog2 übergeben
p1 tee dat p2	stdout von p1 in dat speichern und weiter an p2

Befehlssubstitution

<code>`cmd`</code>	cmd durch stdout ersetzen, oder:
<code>\$(cmd)</code>	Kommando ausführen und Ergebnis einsetzen
	z.B. <code>X=\$(expr \$X + 1)</code>
<code>\$((8-2) / 3)</code>	Rechenausdruck
Bsp: <code>echo "Heute ist der \$(date +%j). Tag im Jahr"</code>	

Variablen

<code>VAR="Ein Test"</code>	<code>VAR</code> definieren und belegen (Keine Leerz.!)
<code>BLAU=\${e[35m]}</code>	Variable mit Kontrollcodes belegen (ANSI, s.u.)
<code>echo \$VAR</code>	Inhalt einer Variablen anzeigen lassen oder: <code>\${VAR}</code>
<code>set</code>	Ausgabe aller Variablen
<code>env</code>	Ausgabe aller Variablen, die weitervererbt werden
<code>export VAR</code>	Variable vererben an aufgerufene Programme
<code>unset VAR</code>	Variable löschen
Wichtige Variablen: HOME, PATH, PWD, PS1, PS2, ? (echo \$?)	

bash: Tipps

<code>cat << END</code>	Einen Text als stdin mitgeben
<code>.... ...</code>	Variablen werden interpretiert!
<code>END</code>	
<code>cat <<< END</code>	Einrückungen erlauben!
<code>....</code>	
<code>END</code>	
<code>echo "7\ nq" ed datei</code>	Bestimmte Zeile anzeigen

Linux-Tools

Tools mit "Datenbank"-Funktionen

wc: Zählt Anzahl Zeilen, Wörter und Zeichen

- l lines: Anzahl Zeilen
- w words: Anzahl Wörter
- c chars: Anzahl Bytes
- m Anzahl Zeichen (UTF-8)
- L max. Zeilenlänge

grep: Alle Zeilen zeigen, auf die das Suchmuster zutrifft

- n Vor jeder Zeile wird die Zeilennummer angezeigt
- c Nur die Anzahl der Zeilen anzeigen
- color Farbiges Markieren der Zeichen (s. Tipp unten)
- v Negation: Alle Zeilen, nur nicht die mit dem Suchmuster ganzes Wort suchen
- w Nur die Dateinamen bei erfolgreicher Suche anzeigen
- e mehrere Suchwörter angeben: -e doz -e stud -e anz
- i Groß-/Kleinschreibung ignorieren
- f datei Suchwörter aus der Datei lesen

Tipp: export GREP_COLOR="5;30;43"; alias grep="grep --color"

cut: Aus einer Datei Spalten/Felder (1,2,...) herausschneiden

- c3-20 Nur bestimmte Spalten anzeigen
- d ";" -f1,3-5 Felder 1,3-5 bei Dateien mit Trennzchn anzeigen
- s suppress: Unterdrücken von Zeilen ohne Feldtrenner

sort: Sortieren von Dateien

- r reverse: absteigende Sortierung
- t ";" -k 3 key: nach 3. Feld sortieren
- t ";" -k 2,5 von Feld 2 bis Feld 5 sortieren
- n Numerisch sortieren
- b Mehrfache Blanks für Sortieren zur einem reduzieren
- u unique: Doppelte Zeilen unterdrücken

join: Führt zwei Dateien anhand eines Index' zusammen

Syntax: join [options] file1 file2

- j n dat1 dat2 bzgl. Feld n verknüpfen
- a dat1 dat2 Von Dateien auch nicht passenden Zeilen
- t ";" -j 3 ... bzgl. Feld 3 verknüpfen
- t ";" -j2 3 In 2. Datei bzgl. Feld 3 verknüpfen (geht auch mit -j1)

head: Gibt die ersten (default: 10) Zeilen aus

- n 20 Die ersten 20 Zeilen

tail: Gibt die letzten (default: 10) Zeilen aus

- n 5 Die letzten 5 Zeilen
- f Dateieinde aktualisiert anzeigen lassen

tr: Transformation von Zeichen, Syntax: tr [optionen] "string1" "string2"

Optionen:

- d Alle angegebenen Zeichen löschen
- s Alle identischen Zeichen zu einem reduzieren
- c Genau die Zeichen nehmen, die nicht angegeben wurden

Bsp: ls -l | tr "a-zäöü" "A-ZÄÖÜ"

nl: Numeriert die Zeilen (nl --help)

Bsp.: ls -l | nl

Regular Expressions

.	belieb. Zeichen	^	Zeilenanfang
*	0-n Wied. letztes Zchn	\$	Zeilenende
+	1-n Wied. letztes Zchn	[a-zä]	Bereich (hier: a-z + ä)
	Oder-Verknüpfung	[^A-Z]	negierter Bereich
\	Maskierung des nächsten Zeichens		

Bsp:

- ^[A-Z] mit Großbuchstaben am Anfang der Zeile beginnend
- ^[a-z]*\$ Alle Zeilen, in denen nur Kleinbuchstaben stehen
- ^[^A-Z] Alle Zeilen, die nicht mit einem Großbst beginnen
- \\. \$ die mit einem Punkt enden
- [Oo]tto die Otto oder otto enthalten.
- [1-30] die die Ziffern 0, 1, 2, 3 enthalten
- ^[^:]*:abc deren zweite Spalte (Trenner :) mit abc beginnt.

Tools zur Shell-Programmierung

expr: Ausdrücke berechnen

- Vergleiche: = > >= < <= !=
- Rechenarten: + - * / %
- Logische Vergleiche: & (UND) | (ODER)
- Funktionen: length, substr, index

cat dateien Dateien ausgeben

- v Nur lesbare Zeichen erzeugen
- n Nummerieren der Zeilen
- s silent. Keine Fehlermeldungen

basename Liefert den Dateinamen aus einem absoluten Pfadnamen und löscht auf Wunsch (2. Parameter) die Endung.

basename /users/men/test.c liefert: test.c

basename /users/men/test.c .c liefert: test

dirname Liefert den Verzeichnisnamen

dirname /users/men/test.c liefert: /users/men

which: Sucht den Pfad einer Programmdatei anhand von PATH.

Bsp.: which lp; which ksh

file: Klassifikation von Dateien (klappt nicht immer genau)

Bsp.: file *

paste: Zusammenfügen von Zeilen (evtl. aus verschiedenen Dateien)

- Steht für eine Spalte in der Standardausgabe
- dat1 dat2 ... Fügt Dateien zusammen

find startdirectory Suche nach Dateien im Dateisystem

Optionen:

- name "*.c" Suche nach angegebenen Dateinamen
- type d Dateitypen: d = Directory, f = normale Dateien
- atime -7 Dateien mit Zugriff in den letzten 7 Tagen
- mtime -20 Geänderte Dateien (in den letzten 20 Tagen)
- user name Alle Dateien eines Users
- maxdepth 3 Maximale Suchtiefe 3 im Verzeichnisbaum
- exec befehl ... {} \; Befehl für Dateien ausführen
- perm oktal Dateien mit bestimmter Zugriffsberechtigung
- size +n Dateien mit bestimmter Größe in Blöcken

showtable -d ":" /etc/passwd Als Tabelle anzeigen

-ht Als HTML-Tabelle (viele weitere Optionen)

Eingabe

read var Zeile von stdin lesen in Shell-Variablen abspeichern

read var1 var2 var3 Einlesen einer Zeile, erstes Wort in var1 abspeichern, zweites in var2, Rest der Zeile in var3. Feldtrenner kann mit IFS vereinbart werden.

IFS=":" Setzt als Trennzeichen den Doppelpunkt statt Space/Tab

Spezial-Tools

sed "Kommandos" datei Stream Editor

- f dname Befehle aus Datei lesen
- n no output

Kommandos: p print, d delete, s substitute

y/abc/ABC/ transform a → A, b → B, c → C

Bsp:

```
-> cat unix.txt | sed "s/Unix/TuNix/g"
-> cat unix.sed
2,3 s/e/#####/
s/Unix/ U n i x /g
-> sed -f unix.sed unix.txt
-> sed -n "[Oo]tto(!)p" dat (=grep (-v) "[Oo]tto" dat)
-> sed -e "s/unix/UNIX/g" -e "/^$/d" dat
-> sed 's/<[^>]+/ /g' HTML-Codes entfernen
-> sed "4 s,'.*',10.0.0.1/8'," in 4. Zeile '*' ersetzen
-> sed -n "2,5!p" -e "s/^\^* /" dat
-> sed "/^$/s/^\^* /g" dat
```

awk 'Kommandos' datei awk-Textverarbeitung

awk [-Ffs] [-v var=value] [program | -f progfile ...] [file ...]

-F ":" Vereinbart als Trennzeichen das Zeichen ":"

(Voreinstellung: Leerzeichen und Tabulatoren, führende Leerzeichen oder Tabulatoren werden ignoriert. Bei Vereinbarung von Trennzeichen werden führende Trennzeichen nicht mehr ignoriert.)

- f awkscript AWK-Script steht in der angegebenen Datei
- v var=wert Vereinbarung einer Variablen.

Bsp.: awk -v HOME=\$HOME

awk-Programmstruktur:

```
BEGIN { <Anweisungen vor Dateilauf> ... }
{ <aktion für alle Datensätze> }

/ muster /
/ muster / { <aktion für best. Datensätze> [ ; <aktion> ... ] }
END { <Anweisungen nach Dateilauf> ... }
```

awk-Variablen

- FS Field-Separator (Datenfeld-Trennung) default: ' ', '\t'
- RS Row-Separator (Datensatz-Trennung) default: '\n'
- OFS Output-Field-Separator
- ORS Output-Record-Separator
- NF Number of Fields (Anzahl Felder im aktuellen Satz)
- NR Number of Records (Anzahl bearbeiteter Datensätze)
- \$0 Aktuelle Zeile
- \$1-\$n Felder des aktuellen Satzes

Anweisungen:

```
if (condition) statement [ else statement ]
while (condition) statement
do statement while (condition)
for (expr1; expr2; expr3) statement
for (var in array) statement
```

Suchmuster:

/^ab/	Alle Zeilen mit "ab" am Anfang
/ab\$/	Alle Zeilen mit "ab" am Ende
/^\$/	Alle Leerzeilen
/[a-z]+/	Ein oder mehrere Kleinbuchstaben
length(\$0) < 50	Alle Zeilen mit weniger als 50 Zeichen
\$5 ~ /Stefan/	Zeilen bei denen im 5. Feld „Stefan“ vorkommt
/Kai/, /Susi/	Alle Zeilen zwischen Kai und Susi (Sortierung!)

Die Suchmuster können mit "&&", "| " und "! " verknüpft werden!

Shell-Programmierung

Shell-Programme starten

```
sh test.sh      Mit Kommandointerpreter starten
. test.sh      in aktueller Umgebung (Variablen, Verz.) starten
chmod u+x test.sh ausführbar machen
./test.sh      im aktuellen Verzeichnis starten
sh -vx test.sh Debuggen
```

Parameter, Optionen

```
$0      Name des Programms selbst
$1...$9 Namen der ersten 9 Parameter (sonst: ${23} )
$*      Inhalte aller Parameter
#$      Anzahl aller Parameter
$$      Prozessnummer
set wort1 wort2 ... Wörter als Parameter setzen ($1, $2, ....)
shift   Parameter nach links shiften ($1 löschen)
set -x   Befehlsprotokollierung ein
```

Rechnen

```
expr \ ( 7 "*" 3 \ ) / 5
    Älteres Rechenprogramm, Entwertungen u. Leerzchn. beachten!
echo ${ausdruck} oder echo ${{ausdruck}}
    Ausdruck berechnen und Ergebnis anzeigen (wie in C)
X=$(( $x * 7 + 5 )) Zeichen gelten als entwertet wie mit "..."!
typeset -i sum=0 sum als Integer-Variablen definieren und mit 0
    initialisieren. Dann funktioniert: sum=$((sum + 4 * 5))
bc      Großes Floatzahlen-Rechen-Tool, Nachkommastellen mit scale einstellen!
Bsp:   echo "2.345 * 1.1111" | bc
        echo "scale=10; 2.345 * 1.1111" | bc
seq 1 10 Zahlen von 1-10 generieren, auch: echo {1..9}
```

Bedingungen

```
test ausdruck. bzw. [ ausdruck ]
-f datei      Datei existiert und ist eine normale Datei
-d datei      Datei existiert und ist ein Directory
-s datei      Datei existiert und ist nicht leer

"string"     Zeichenkette ist nicht leer
-z "string"   Zeichenkette ist leer
str1 == str2  Zeichenketten sind identisch (alt: "=")
str1 != str2  Zeichenketten sind nicht identisch

z1 -eq z2     Zahlgleichheit -eq (=) -ne (!=)
z1 -le z2     less equal/than -le (<=) -lt (<)
z1 -ge z2     greater equal/than -ge (>=) -gt (>)

exp1 -a exp2 AND-Verknüpfung von Ausdrücken
exp1 -o exp2 OR-Verknüpfung von Ausdrücken
! ausdruck   Negation
```

Variablen verwenden

```
Name=$(echo HansWurst | cut -c1-4)
read var      Variable von Tastatur belegen
jahr=2016; cal='cal $jahr';
eval $cal     Indirekte Variablenauswertung
test "$VAR" -eq "$VAR" Test auf numerisch
```

Felder:

```
feld=(Menne Stefan Paderborn) Feld vereinbaren
declare -a feld=(Menne Stefan Paderborn)
echo ${feld[0]} Erstes Element anzeigen
echo ${feld[*]} Inhalte aller Felder ausgeben
```

Variablen-Operationen:

```
${VAR:5}      Alle Zeichen von VAR ab 6. Zeichen
${VAR:0:4}    Vier Zeichen vom Anfang (= 0 !!)
${VAR="default Wert"} Leer od. NichtDef: Default-Wert
${VAR+"VAR ist definiert!"} Var ex.: Text anzeigen
${VAR-"defaultwert"} Default-Wert definieren
${VAR?"ERR: VAR nicht def."} Var ex. nicht: Text anzeigen
${VAR#prefix} Anfang des Var-Inh. entfernen
(## alles entfernen, Bsp: ${VAR##*/} wirkt wie "basename")
${VAR%suffix} Suffix entfernen
Bsp: ${PFAD%/*} letztes "/" + weitere Zchn. entfernen
${PFAD%/*/*} nach erstem "/" alles entfernen
${VAR//alt/neu} Textersetzung
${#VAR}      Länge der Variablen (auch Arrays)
```

Bildschirmsteuerung

```
tput parameter: Bildschirmsteuerung, Bsp. für Parameter:
clear      Bildschirm löschen, home Cursor nach links oben
civis     Cursor unsichtbar cvvis Cursor sichtbar
lines     Anz. Bildschirmzeilen cols Anzahl Spalten
cup 2 30  2. Zeile, 30. Spalte bold Fett
smso      Invers blink blinkend
smul      unterstrichen sgr0 Sonderschrift aus
```

Verzweigungen

```
if test ! "$1"
then
echo "usage: $0 <parameter>!"
exit 1
fi

if grep menne * 2> /dev/null
then
echo "---ENDE---"; exit 0
else
echo "---Nichts---"; exit 1
fi

if test -d "$1" ;then
echo "$1: Directory!"
elif test -f "$1" ;then
echo "$1: normale Datei!"
else echo "$1: fehlt od. sonstige "
fi

case auswahl in
ende) echo Ende; exit 0
;;
anfang|begin) echo Anfang
;;
*) echo "Falsche Wahl"
esac
```

Schleifen

```
for datei in *
do
zeilen=$(wc -l < "$datei")
echo "$datei: $zeilen"
done

for (( i=30; $i < 50; i++ ))
do
echo -e "\e[0m\e[${i}]m Farbe: $i"
done

ls -l | while read zeile
do
ZUGRECHTE=${zeile:0:8}
NAME=${zeile:60}
echo "$ZUGRECHTE:$NAME"
done

for par in $*
do
echo "Parameter: $par"
done

"gehalt.dat":
Müller:12:3456
Meyer:13:3432
Menne:13:4321

"gehalt.sh":
IFS=":" # für "read"
typeset -i sum=0 mgeh
while read name monate gehalt
do
mg="$monate * $gehalt"
sum=$((sum + $mgeh))
printf "%-9s: %8d\n" $nam $mg
done < gehalt.dat
printf "%-9s: %8d\n" "Ges:" $sum
```

Kontrollanweisungen:

```
break      Schleifenabbruch: Sprung nach done
continue   Schleifenfortsetzung: Sprung vor done
```

Funktionen und Prozeduren

Müssen vor dem eigentlichen Shell-Programm definiert werden.
Prozeduren: Funktion mit Rückgabewert:

```
function ausfuehrbar
{
if test -x $1
then echo "$1 ausführbar"
else echo "$1 nicht ausführbar"
fi
}

is_groesser ()
{
if test $1 -le $2
then return 0 # TRUE
else return 1 # FALSE
fi
}

Aufruf:
-> ausfuehrbar prog.x
prog.x ist ausführbar!
->

Aufruf:
-> groesser 23 6
-> echo $?
1
```

Tipps

ANSI-Escape-Sequenzen:

```
echo -en "\ec"      Bildschirm löschen
echo -en "\e[J"    Zeile bis Ende löschen
echo -en "\e[35m"  Farbnummer 35 setzen (sinnvoll: 30-49)
echo -en "\e[0m"   Farbe zurücksetzen
echo -e "\e[2;50H" Cursor in Zeile 2, Spalte 50 (Nummerierung ab 1)
```

Diese Referenz ist im Rahmen meiner Unterrichtstätigkeit am bib entstanden. Die Anordnung und Auswahl der Befehle ist aus meinen persönlichen Bedürfnissen für unterrichtliche Zwecke entstanden.

Aufgrund der vielen Nachfragen stelle ich die Referenz seit 12/2001 ins Internet. Die Seiten 1-4 sind die wesentlichen Seiten.

Verwenden Sie diese Referenz als Nachschlagewerk für Linux-Befehle. Jede Seite darf frei für Lehrveranstaltungen kopiert und verteilt werden, wenn mein Copyright unten auf der Fußzeile lesbar bleibt.

Anregungen und Meinungen schreiben Sie bitte an:
Stefan.Menne@gmx.de

Papierkorb:

ypcat passwd	<i>Userdatei anzeigen lassen (NIS)</i>
ypmatch dozmen passwd	<i>Usereintrag anzeigen lassen (NIS)</i>