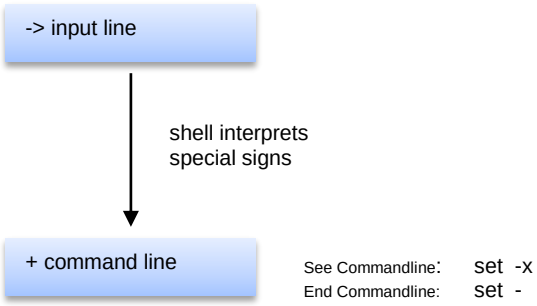
Unix

Reference

6.6

Special Signs:

Whitespaces:	SPACE \t \n
Wildcards:	? * [...]
I/O-Redirection:	< > << >> |
Commands:	&& || ; (...)
Shell-Variable:	$ ${...}
Substitution:	`...` $(...) $((...))
Special-Signs:	"..." '...' \



Unix Basics

Command Line

⇧⇩	cursor: show last commands
⇥	tab: complete command- and file-names
^C	abort the program
^D	sign for end of text

Directory Commands

ls list directory on the screen

-l	long listing	-a	all files (also hidden)
-F	display filetype	-i	show inodes
-d	list directory itself	-R	recursive (all subdirectorys)

cd *directory* change to an other directory

pwd print working directory)

mkdir *directory* make directory

-p create with complete path

rmdir *directory* remove directory (if empty)

rm -rf *directory* kill complete directory

Directory Symbols:

~	home-directory (like $HOME, e.g.: ls ~menne/Public)
/	root-directory (ls /)
.	link to the working directory
. .	link to the upper directory

Directory/Filename Symbols (Joker|Wildcards):

*	any amount of chars	e.g.:	cp * ..
?	exactly one char	e.g.:	more test?.txt
[a-z]	a char from a to z	e.g.:	vi [a-d]*.c
[!a-z]	no char of this block	e.g.:	ls [!a-z]*

File Commands

more *file* Show File Content page by page

cat *file* Show (complete) File Content

-v Show special signs

rm *files* **remove** file/directorytrees

-R recursive erase (-r) -i interrogate before erase

-f force: erase also protected files

cp *files directory* | **cp** *file file*

copy files into a directory or duplicate one file

-R recursive kopieren -i interrogate before replacing

mv *files directory* | **mv** *file file*

move files to a directory or rename a file

ln -s *filepfad directory* create a symbolic link

ln *von nach* create a hard link

touch *files* create a file or change access date

lp *file* print file

Masquerade of Signs

If you want to avoid the special meaning of signs like: *, ?, &, ;, <, > ... :

\$	Use \ do avoid the meaning of the following sign
'...'	Masquerade all signs, also $ and \
"..."	Masquerade common signs, except $ and \

Output

echo show all parameters, divided by a blank-character

echo -e bash: interpretation of special signs

echo -n suppress newline

\a	tone	\b	backspace
\r	beginning of line	\t	tabulator
\n	newline	\033	oktal codes

e.g.: echo -e "Menne: \t 1234"

printf print formatted like in C:

%20s string with 20 characters right-aligned (also "%20.10s")

%-20s string left-aligned with 20 chars

%8d digits with 8 chars

e.g.: printf "%-20s : %4d,- Euro\n" "Menne“ 1234

Information

man *command* show manual for command (z.B. man ls)

(„*space*“ for next page, „q“ for end, „h“ for help)

-k *searchterm* show commands for the search term

man ksh | col -bx show manual without formatting

date show current date and time

many formattings possible:

date +%d.%m.%Y	german date (=%x)
date +%H:%M:%S	time (=%X)
date "+%j-ter Tag %A"	day of year and week date
date -d "2000-08-03" +%j	calculate day of year for a date
date -d "-45 days"	45 days before

Date-Difference:

expr \($(date -d "2018-05-24" +%s) - $(date +%s) \) / 86400

Special Signs

space, tab	separator for options (also newline possible!)
~	path of home directory (also: "~menne")
;	separator for commands
(....)	combination of commands: (ls; who) | more
&&	case of success: grep men * && echo "ok"
||	case of error: grep men * || echo "error"

Special Unix Commands

clear clear screen

cal 1 2000 show calendar (beginning with monday: cal -ym)

who who is logged in

vi -x test.txt edit with crypting the file

quota -v show the used disk-space

du -sh ~/* disk usage mit sum and human readable

find ~ -user yyy -size +1000k -exec `ll` {} \; look für big files

mail –s „*Hallo***“** men@bib.de **<** *letter.txt* send an email

script create a transcript for the screen listings

history show the last commands

time *command* timing of a command (Hint: export TIMEFORMAT="%R")

Stop Process:

ps list of processes (auch: **top**)

-e of every user (Linux: -x)

-f full listing -l long listing

kill -9 *pid* kill a process (-9: process cannot ignore it)

Abbreviation:

alias ll="ls -l“ create an abbreviation

unalias *name* kill an abbreviation

Change the Code:

recode pc < windows.txt	code the newline-character
recode /cr-lf..latin1 <windows.txt	recode a windows file
recode latin-1..UTF-8 file	recode a windows file to UTF-8

Bash-Configuration

export LANG=en_EN.UTF-8 english language setting

set -o show all options

set +o histexpand kill an option (e.g.: echo „hello!“)

~/.bash_profile personal file with login commands

~/.bash_logout personal file with logout commands

~/.exrc file for configuration of the vi

~/.sh_history file with the last commands

Access Rights

Files:		***Directorys:***	
r	*readable*	*x*	*enter (cd is possible)*
w	*writeable*	*r*	*readable (ls, find)*
x	*executable*	*w*	*files/dirs (create or delete files)*

chmod *usergroup*[+|-|=]*rights files*

(u=user, g=group, o=others, a=all) (+ add, - revoke, = absolute)

e.g..: chmod 751 *.sh *.cmd ; chmod og+rx *.sh

(StickyBits: u+s, g+s, o+t or e.g. chmod 4744 .)

I/O Redirection

channels: 0 stdin, 1 stdout, 2 stderr

< *file*	stdin out of file
1> *file*	stdout into file
1>> *file*	add stdout to the end of the file
2> *file*	stderr into file
2>&1 > *file*	stdout and stderr together into file
2> /dev/null	stderr to the trash
cmd1 | *cmd2*	give stdout of command1 to stdin of command2
p1 | tee *dat* | *p2*	store stdout of cmd1 into dat and give it also to cmd2

Substitution for commands:

`*cmd*`	replace command with its stdout, newer alternative:
$(*cmd*)	replace command with stdout e.g. X=$(expr $X + 1)

Shell Variables

set	show all variables and functions
env	show the environment for each new process
VAR=test	define VAR with value (**NO expansion space!**)
BLAU=$'\e[35m'	define variable with control codes (ANSI codes)
echo $VAR	show value of a variable, also: ${VAR}
export VAR	set a variable into the environment (export VAR=xxx)
unset VAR	erase a variable
urgent variables:	HOME, PATH, PWD, PS1, PS2, ? (echo $?)

ksh, bash: Hints

cat << END here script

....<a href=$link> ... (with interpretation of variables!)

END

cat <<- END allow indentations!

....

END

echo "7\nq" | ed file show an special line

ypcat passwd show file of users in network (NIS)

ypmatch menne passwd show an special user (NIS)

Unix- Tools

Tools With Database Functions

wc: word counter**,** count lines, chars and word
-l show only #lines -w amount of words
-c show only #chars -m amount of UTF-8-chars
-L maximum length of a line

grep *searchterm files*: show the lines with a search term
-n with line numbers
-c only amount of lines
--color coloration of the search term (*see hint below*)
-v negation: show lines except thos with the search term
-w seach for a whole word
-l show only the file names at success of search
-e multiple search terms: -e doz -e stud -e anz
-i ignore upper and lower case
-f *file* read the search terms out of a file
hint: export GREP_COLOR="5;30;43"; alias grep="grep --color"

cut: cut columns or fields out of a file
–c3-20 columns 3 until 20
–d ":" use ":" as a separator of fields
–f1,3-5 show fields 1 and 3 until 5
–s suppress empty lines

sort: sort content
-r reverse order
-t ";" use „;“ as separator of fields
-k n[,m] sort from field n (to field m)
-n sort numerically (12 ist more than 3)
-b reduce multiple blanks to one only
-u suppress multiple identical lines

join: join two files with a key expression
syntax: join [options] file1 file2
-a *n* show also lines without the key expression from file n
-j *n* join with key expression in field n
-2 *n* join on file2 with key expression in field n
-t *ch* define the field separator

head: show the first lines (default: 10)
-n *number* amount of lines

tail: show the last lines (default: 10)
-n *number* amount of lines
-f continous looking at the end of the file

tr: transformation of characters
syntax: tr [options] "string1" "string2"
options:
-d delete the characters, e.g. tr -d “[a-z][A-Z]”
-s strip identical characters to one only, e.g. tr -s “aeiou”
-c change the characters to the opposite
e.g.: ls –l | tr "a-zäöü" "A-ZÄÖÜ"

Regular Expressions

.	every char	^	beginning of line
*	0-n repetitions of a char	$	end of line
+	1-n repetition of last char	[a-z]	area of characters
|	or combination	[^A-Z]	not this area of chars
\	masking the next character		

e.g.:
^[A-Z] all lines with a capital letter at beginning of the line
^[a-z]*$ all lines which completely consist of lowercase characters
^[^A-Z] all lines which do not begin with a capital letter
\.$ all lines which end with a dot
[Oo]tto all lines with Otto or otto.
[1-30] all lines with numbers like 0, 1, 2, 3
^[^:]*:abc all lines which begin in the second field with abc (sep. ":")

Tools for Shell-Scripting

expr: calculate expressions
1. comparisons: = > >= < <= !=
2. calculation + - * / %
3. logical comparisons & (AND) | (OR)
4. Functions: length, substr, index

cat *files* concatenate files
-v generate readable characters (make special signs readable)
-n numbering the lines
-s silent: no error messages

basename shows the basename of an path or strip the end
basename /home/men/test.c shows: test.c
basename /home/men/test.c .c shows: test

dirname shows the directory name of an path
dirname /home/men/test.c shows: /home/men

which: looks for the exact path of an command
e.g..: which lp
which ksh

nl: numbering the lines
syntax: nl [options] file

file: classification of an file
e.g..: file *

paste *file1 file2 file3 ….* **:** merge lines of files to columns
– stand for a line of stdin, e.g.: ls | paste – – –
dat1 dat2 ... merge the files to columns

find *startdirectory* search for files in that directory
options:
-name "*.c“ look for files like “*.c”
-type *d* file type: d = directory, f = normal files
-atime -7 files with access time in the last week
-mtime -20 files with modification time in the last 20 days
-user *name* all files of an user
-maxdepth 3 maximum search depth in the directory tree
-exec *command.. {}* \; execute the command at every file
-perm *octal* files with certain octal permissions (e.g. 644)
-size *+n* files with a certain size

read *var* read a line from stdin and store it in variable „var“
read *var1 var2 var3*: read a line form stdin, first word in *var1* second in *var2*, rest of line in *var3*.
set the field separator with variable **IFS:**
IFS=":“ set the input field separator to „:“ in spite of space/tab

Special-Tools

showtable -d ":" /etc/passwd show the file as a table
-ht show it as HTML, *many other options!*

sed *"commands" file* stream editor
-f *dname* read the command out of a file
-n no output
commands: p print, d delete, s substitute
y/abc/ABC/ transform a → A, b → B, c → C
e.g.:

```
-> 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 "/^$/d" dat
-> sed 's/<[^>]\+//g' # erase HTML-codes
-> sed -n "2,5!p" -e "s/^/\* /" dat
-> sed ‘’/^$/!s/^/# /g’’ dat
```

awk ‘commands’ file awk-text manipulation
awk [-Ffs] [-v var=value] [program | -f progfile ...] [file ...]
-F ":" define as separator the sign ":"
(default: space and tab, leading spaces and tabs are ignored. If you define an separator with –F, leading spaces and tabs are NOT ignored)
-f awkscript read the awk-script out of a file
-v var=wert define a variable
e.g..: awk -v HOME=$HOME

awk program structure:

BEGIN	{ *<command before script>* ....]}
	{ *<activity for all lines >* }
/ muster /	
/ muster /	{ *<activity for special lines >* [; *< activity>* ....]}
/ muster /	{ *<activity>*
	< activity>
	}
END	{ *< command after script >* ....]}

awk-variables:

FS	field- separator	default: ' ', '\t'
RS	row- separator	default: '\n' (= line)
OFS	output- field- separator	
ORS	output- record-separator	
NF	number of fields of the current record	
NR	number of rows	
$0	current line	
$1-$n	fields of the current line	

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

regular expressions for awk

/^ab/	all lines with "ab" at the beginning
/ab$/	all lines with "ab" at the end
/^$/	all empty lines
/[a-z]+/	a line with one or more lowercase letters
length($0) < 50	all lines with fewer than 50 characters
$5 ~ /Stefan/	all lines within “Stefan” in the fifth field
/Meier/, /Menne/	all lines between Meier and Menne (look out for the sorting order!)

you can combine them with "&&", "||" and "!"

Shell-Scripting

Start Shell-Scripts

a) make a file executable:	`chmod u+x test.sh`
start it in current directory:	`./test.sh`
b) start it in current environement:	`. test.sh`
c) start it with an special shell:	`sh test.sh`
d) for debugging:	`sh -vx test.sh`

Parameter, Options

$0	name of the script itself
$1...$9	the first nine parameters (otherwise: ${23})
$*	content of all parameters
$#	number of the parameters
$$	number of current process
set *word1 word2* **...**	set words as parameters ($1, $2, ...)
shift	left shift (here: erase of $1, $1 has content of $2)
set -x	show the commands before executing

Calculation

expr \(7 "*" 3 \) / 5
older programm to calculate, beware of spaces and masquerading!

$[*expression***]** or **$((***expression***))**
calculate the expression and show the result (like in C)
e.g. X=$[$x * 7 + 5] all signs are masked with "..."

typeset –i sum=0 declare „sum“ as a integer variable and initialize it with Zero. Then ist possible: sum="$sum + 4 * 5"

bc great float calculation tool, set decimal places with scale
e.g.: echo "2.345 * 1.1111" | bc
echo "scale=10; 2.345 * 1.1111" | bc

seq 1 10 generate numbers from 1 to 10

Conditions

test *expression.* alternative: **[** *expression***]**

–f *file*	file exists and is a normal file			
–d *file*	file exists and is a directory			
–s *file*	file exists and is not empty			
"*string*"	string is not empty			
-z "*string*"	string is empty			
str1 == *str2*	strings are identical (old: "=")			
str1 != *str2*	strings are not identical			
z1 -eq *z2*	equal numbers	-eq (=)	-ne (!=)	
z1 -le *z2*	less equal/than	-le (<=)	-lt (<)	
z1 -ge *z2*	greater equal/than	-ge (>=)	-gt (>)	
exp1 -a *exp2*	AND combination of expressions			
exp1 -o *exp2*	OR combination of expressions			
! *ausdruck*	Negation			

Usage of Variables

```
Name=$(echo HansWurst | cut -c1-4)
read var                                read a variable from stdin
eval echo "\$$variable"                 indirect evaluation of a variable
                or VAR=HOME ; d="echo $VAR"; eval $d
test for numerical: test "$VAR" -eq "$VAR"  2> /dev/null
```

Arrays:

```
field=(Menne Stefan Paderborn)      declare an array
    declare –a field=(Menne Stefan Paderborn)
echo ${field[0]}                    show first element
echo ${field[*]}                    show all elements
```

Operations on Variables:

```
${VAR:5}                            show conten beginning at 6th sign
${VAR:0:4}                          show four chas from beginning (=0!)
${VAR="default content"}            if empty or not defined: give default
${VAR+"VAR is defined!"}            if VAR exists: show text
${VAR-"defaultwert"}                define an default content
${VAR?"ERR: VAR not def."}          Var exists not: show text
```

${VAR#prefix} strip the beginning of content
(## erase all, e.g.: ${VAR##*/} ist like "basename")
${VAR%suffix} erase the suffix
e.g.: ${PFAD%/*} erase last "/" and all afterwards
${PFAD%%/*} erase all after first "/"
${VAR/old/new} replace text
${#VAR} length of a variable or array

Branching

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

```
if grep menne * 2> /dev/null
then echo "---END---"; exit 0
else echo "---Nothing---"; exit
1
fi
```

```
if test -d "$1" ;then
  echo "$1: directory!"
elif test -f "$1" ;then
  echo "$1: normal file!"
else echo "$1: dont know "
fi
```

```
case select in
    end)    echo End; exit 0
            ;;
    anfang|begin) echo Beginning
            ;;
    *)      echo "Wrong Choice"
esac
```

Repetitions

```
for file in *
do
  lines=$(wc –l < "$file")
  echo "$file: $lines"
done
```

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

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

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

```
"earnings.dat":
   Müller:12:3456
   Meyer:13:3432
   Menne:13:4321
"earnings.sh":
IFS=":"      # for "read"
typeset –i sum=0 mg
while read name months
earning
do
  mg="$months * $earning "
  sum="$sum + $mg"
  printf "%-9s: %8d\n" $name
$mg
done < earnings.dat
printf "%-9s: %8d\n" "Total:" $sum
```

***Controls*:**

break	go after word done
continue	continue the repetition with next word

functions

must be declared before the script.

functions:

```
function executeable
{ if test -x $1
  then echo "$1 is executeable "
  else echo "$1 not executeable"
  fi
}

Call:
-> executeable prog.x
prog.x is executeable
->
```

functions with retun value:

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

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

Hints

Controls For The Screen

tput *parameter*
Parameters:

clear	*clear screen*,	*home*	cursor left corner above
civis	unvisable cursor	*cvvis*	cursor is visable
lines	amount of lines	*cols*	amount of columns
cup 2 30	2. line, 30. column	*bold*	bold letters
smso	invers letters	*blink*	blinking
smul	underline	*sgr0*	no special colors

ANSI-Escape-Sequenzes:

echo -en "\ec"	clear screen
echo -en "\e[J"	erase until end of line
echo -en "\e[35m"	set color to 35 (colors from 30 to 49
echo -en "\e[0m"	reset colors
echo -e "\e[2;50H"	cusor to line 2, column 50 (numering start at 1)

This reference was developed for my lessons at the bib international college in Paderborn, Germany.

The order and the selection of the commands is a result of personal choice as it aids my tuition of Unix.

This reference has been made public in 2001 upon request of my students.

You may use this reference for your personal needs, also you can use it for lessons in university or school. But leave my Copyright on the bottom of pages you want to copy.

Wishes an critic you can send to:
Stefan.Menne@gmx.de