Bash

🔧 Fix Me!

CheatSheet: https://devhints.io/bash

root-check

```
if [ $UID -eq 0 ];then echo "root";fi
```

online-check

```
  ping -c 1 ${HOST} -W 1 >/dev/null
  if [ $? -eq 0 ];
  then
    echo "host is alive"
  fi
```

Dateigröße

```
  fs=$(wc -c < file)
  if [ $fs -eq 0 ];
  then
    echo "file is empty"
  fi
```

Massen-Datei-Verarbeitung

```
for file in *.html
do
  mv "$file" "${file%.html}.txt"
  #"${file%.*}.txt"
done

#auch gut, um z.b. von mehreren Dateien die Checksumme zu errechnen und in
gleichnamiger Datei zu hinterlegen
for file in *.gz; do md5sum "$file" >"$file.md5";done

#auch mit der Klammern-Auswertung lassen sich bestimmte Sachen machen
ls *.{php,html}
```

Farben

```
clr_red=$'\e[1;31m'
clr_green=$'\e[1;32m'
clr_yellow=$'\e[1;33m'
clr_blue=$'\e[1;34m'
clr_reset=$'\e[0m'
```

```
 echo ${clr_red}test${clr_reset}
```

logging

einzelnen Befehl mitloggen (nur stderr) und gleichzeitig anzeigen (stdout+stderr)

```
#via Process Substitution
(https://www.gnu.org/software/bash/manual/bash.html#Process-Substitution)
make 2> >( tee "$logfile" ) ; echo $?
```

ganzen Block mitloggen:

```
exec 3> >(tee logfile)

make alpha 2>&3 &&
make bravo 2>&3 &&
make charlie 2>&3 && success=1

exec 3>&-
```

komplettes script mitloggen (mit Anzeige):

```
exec &> >(tee -a "$logfile")
```

Möglichkeiten echo an stderr zu schicken:

```
echoerr() { echo "$@" 1>&2; }
```

```
echoerr hello world
```

https://stackoverflow.com/a/2990533

Debug-Modus

am Anfang des Scripts:

```
test -n "$DEBUG" && set -x
#Anzeige der aktuellen Datei (funktioniert auch wenn ge-source-d)
```

```
test -n "$DEBUG" && echo ${BASH_SOURCE[0]}
```

Aufruf mit:

```
DEBUG=y script.sh
```

ggf. mit PS4 die Ausgabe anpassen (Quelle)

```
PS4='Line $LINENO @ $(date +%s.%N): ' bash -x script
```

vorherige Instanzen beenden

```
#kill previous instances of current script
script_name=${BASH_SOURCE[0]}
for pid in $(pidof -x $script_name); do
  if [ $pid != $$ ]; then
    kill -9 $pid
  fi
done
```

https://unix.stackexchange.com/a/213293

Case

```
case "$string" in
  *foo*)
    # irgendwas
  ;;
  [1-6]*)
    #zahlenbereiche
  ;;
  "2" | "3")
  multiple values
  ;;
  *)
    #alles andere
  ;;
esac
```

fallthrough (;&) and resume (;;&) https://stackoverflow.com/a/24544780

arrays

```
arr=(a b c d e f g h)
arr[8]="i"
unset arr[2] #entfernt 3. Element
echo ${arr[@]} #ganzes Array anzeigen
```

```
echo ${#arr[@]} #Anzahl der Elemente
arr+=('foo') #Element anhängen
```

Eingabe mit Vorbelegung

```
  name="uImage"
  read -e -i "$name" -p "Please enter filename: " input
  name="${input:-$name}"
```

Stringmanipulation

In Bash 4: alles klein

```
$ echo "${string,,}"
```

alles groß:

```
$ echo "${string^^}"
```

Quelle

Pfad / Verzeichnis

```
$ VAR=/home/me/mydir/file.c
$ DIR=$(dirname "${VAR}")
$ echo "${DIR}"
/home/me/mydir
$ basename "${VAR}"
file.c
```

```
~% FILE="example.tar.gz"
~% echo "${FILE%%.*}"
example
~% echo "${FILE%.*}"
example.tar
~% echo "${FILE#*.}"
tar.gz
~% echo "${FILE##*.}"
gz
```

mehr infos hier

Teilstrings

```
a=string
```

```
b=${a:p:l}
#p=Position (0-basiert),l=Länge
```

ersetzen

```
orig="AxxBCyyyDEFzzLMN"
mod=${orig//[xyz]/_}
```

Regex-Prüfung

```
string='My string';

if [[ $string =~ .*My.* ]]
then
  echo "It's there!"
fi
```

Split via Trennzeichen

```
OLDIFS="$IFS"
IFS='|' read -r -a array <<< "$str"
#echo "array ("${#array[@]}"): "${array[@]};
#echo "first item: ${array[0]}"
IFS="$OLDIFS"
```

Alternative wenn mehrere Teile direkt weiterverwendet werden sollen (hier Berechnung der nächsten Kernelversion für inkrementellen Patch):

```
k=$(make kernelversion)
#4.4.140
kn=$(echo $k|awk -F. '{print $1"."$2"."($3+1)}')
#4.4.141
kf=$(echo $k|awk -F. '{print $1"."$2"."$3"-"($3+1)}')
#4.4.140-141
```

string mehrfach

```
printf '=%.0s' {1..100};echo
```

https://stackoverflow.com/questions/5349718/how-can-i-repeat-a-character-in-bash

rechnen

```
GPIO_NO=$((232+25))
```

alternativ (z.B. für array-Index)

```
COL=0
id=${array[$[ COL++ ]]}
```

Ausführzeit

```
P1=$(date +%s) #starttime as unix-time
#do something
P2=$(date +%s) #endtime as unixtime
echo $[$P2-$P1] #calculate and print the difference
```

Zahlenbereiche

```
ls IMG_20170923_{17..18}*
```

Sprache

kurzzeitig umstellen (z.b. um Fehlermeldungen auf englisch zu bekommen für Foren):

```
LANG=C
```

Prompt

```
export PS1="[\D{%F %T}] \u@\h\n\w$"
```

in die ~/.bashrc

Farben lassen sich so realisieren

```
\[\033[1;31m\] => rot
\[\033[1;32m\] => grün
\[\033[1;33m\] => gelb
```

https://wiki.ubuntuusers.de/Bash/Prompt/#Farben

LS_COLORS

```
export
LS_COLORS='di=1;32:fi=0:ln=31:pi=5:so=5:bd=5:cd=5:or=31:mi=0:ex=35:*~=90:*.b
ak=90:*.zip=35:*.tar=35:*.gz=35:*.bz2=35'
```

http://www.bigsoft.co.uk/blog/2008/04/11/configuring-ls_colors

Farben temporär deaktivieren

```
\ls
```

Oder

```
dir
```

Zeit-Darstellung

eigene locale für Datum nehmen

```
LC_ALL=de_DE.utf8 date '+%x'
```

eigene Zeitzone

```
export TZ=Europe/Berlin
dt=$(date +"%Y-%m-%d %H:%M:%S")
```

Zeitrechnung

Zeitstempel als String (für Backups)

```
D=$(date +"%Y-%m-%d %H:%M:%S")
```

Zeitrechnung (+1 Tag)

```
DT=$(date -d "+1 day" +"%Y-%m-%d")
```

Umrechnung in Unix-Zeitstempel

```
date -d "2018-06-14 03:00:00" +%s
1528938000
```

Unix-Zeitstempel zurückformatieren

```
date -d "@1528938000" "+%d.%m.%y %H:%M:%S"
14.06.18 03:00:00
```

Berechung (MySQL-Zeitstempel -10 Minuten)

```
sd=$(date -d "2018-06-14 03:00:00" +%s)
sd2=$(($sd-10*60))
date -d "@$sd2" "+%y-%m-%d %H:%M:%S"
18-06-14 02:50:00
```

MySQL in Bash

```
H=host
U=user
P=password
D=database
T=table
LOGFILE=/tmp/script.log

Q="SELECT * FROM $T";
data=$(mysql -h$H -u$U -p$P -D$D -Bse "$Q"  2>&1)
[ $? = 0 ] || echo $data >> $LOGFILE

while read line; do
    echo "Zeile: $line"
    IFS=$'\t' read -ra STR_ARRAY <<< "$line"
    echo "1st col: ${STR_ARRAY[0]}"
    echo "2nd col: ${STR_ARRAY[1]}"
    echo "3rd col: ${STR_ARRAY[2]}"
done <<< "$data"
```

Grep

nach mehreren Strings suchen (ODER)

```
grep 'echo\|then' file.txt
```

regex

```
grep 'e[c][h]o' file.txt
```

SED

suchen ersetzen

```
sed 's/echo/blah/' file.txt
```

```
sed 's/\(echo\)/blah_\1/' file.txt
```

Mehrfach-Ersetzung:

```
sed -e 's/a/b/g ; s/b/d/g' file
sed -ibak -e 's/a/b/g' -e 's/b/d/g' file
```

https://unix.stackexchange.com/questions/268640/make-multiple-edits-with-a-single-call-to-sed#

text zwischen Start- und Endmarkierung ausgeben

```
sed -n -e '/^BEGIN$/,/^END$/{//!p}' file.txt
```

diff

git diff ähnliche Ausgabe

```
diff -urpN <file1|dir1> <file2|dir2>
```

tar

Zielverzeichnis angeben

```
tar -C $dir -xzf $file
```

System-backup

```
tar -cvpzf backup.tar.gz --exclude=backup.tar.gz --one-file-system /
```

Parameter parsen

https://gist.github.com/jehiah/855086

```
#!/bin/bash
LOG=0
TEST=0
usage() {
  echo "$0 allowed options:"
  echo -e '\t -l|--log|--logging\tenable Logging'
  echo -e '\t -t|--test\tenable Testmode'
}
while [ "$1" != "" ]; do
  PARAM=$(echo $1 | awk -F= '{print $1}')
  VALUE=$(echo $1 | sed 's/^[^=]*=//g')
  echo $PARAM"=>"$VALUE
  case $PARAM in
    -l|--log|--logging)
      echo "LOG"
      LOG=1;
    ;;
    -t|--test)
      echo "TEST"
      TEST=1;
    ;;
    *)
      echo "ERROR: unknown parameter \"$PARAM\""
      usage
```

```
      exit 1
    ;;
  esac
  shift
done
```

ip-adresse

```
ip -4 -o addr show enp3s0 | sed 's/^.*inet \(.*\) brd.*$/\1/'
```

find / exec

```
find . -name '*.rej' -exec rm {} \+
```

Dateien umbenennen/an programme geben

```
OLDIFS=$IFS;IFS=$'\n'; for i in *.mp4; do ffmpeg -i "$i"
"${i%.*}.mp3";done;IFS=$OLDIFS;
```

mögliche Alternative (ganz unten wg. Whitespaces):
https://unix.stackexchange.com/questions/114908/bash-script-to-convert-all-flac-to-mp3-with-ffmpeg

alias / oneline function

```
alias t='...'
unalias t
```

```
hxd() { hd "$1" | less; }
```

git info

```
parseGitBranch() {
    git rev-parse --abbrev-ref HEAD 2> /dev/null | sed -e 's/\(.*\)/(\1)/'
}

gitstatus() {
        status=$(git status --short 2> /dev/null)
        if [[ $? -eq 0 ]];then
                modified=$(echo "$status"|grep "^.\?M" | wc -l)
                untracked=$(echo "$status"|grep "??" | wc -l)
                echo " [${modified}M${untracked}U]";
        fi
}
```

```
if [ "$color_prompt" = yes ]; then
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\0
33[01;34m\]\w\[\033[00m\] $(parseGitBranch)$(gitstatus)\n\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w
$(parseGitBranch)$(gitstatus)\n\$ '
fi

unset color_prompt force_color_prompt
```

xterm title

```
trap 'echo -ne "\033]0;$BASH_COMMAND\007"' DEBUG
function show_name(){
    if [[ -n "$BASH_COMMAND" ]];
    then
    echo -en "\033]0;$(basename $(pwd))\007";
    else
    echo -en "\033]0;$BASH_COMMAND\007";
    fi
}
export PROMPT_COMMAND='show_name'
```

https://unix.stackexchange.com/a/91796