Bash: Programmieren, ohne zu programmieren

Holger Klawitter <info@klawitter.de>

12. November 2001

Was ist die Bash?

- Die Bash ist nicht nur eine Eingabehilfe …
- … sondern auch eine Programmierumgebung
- Ein Shellskript ist ersteinmal eine Liste von gespeicherten Befehlen

```
#!/bin/sh
#
# Datei: backup.1
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-08
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

find . -name "*~" -exec rm -f {} ";"
tar -cz --exclude-from=/home/holger/Vortrag/backups/config/alles.ne
in --files-from=/home/holger/Vortrag/backups/config/alles.ja -f /ho
me/holger/Vortrag/backups/data/backup.tgz
```

Das “Drumherum” eines Shellskripts

- Jede Zeile ist ein Befehl.
- Datei muss erreichbar sein.
- Die Datei muss ausführbar sein.
- Die gewünschte Shell sollte angeben werden.
- Skripte werden wie Programme gestartet.

```
#!/bin/sh
#
# Datei: backup.2
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-25
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

backups=/home/holger/Vortrag/backups

find . -name "*~" -exec rm -f {} ";"

tar -cz \
  --exclude-from=$backups/config/alles.nein \
  --files-from=$backups/config/alles.ja \
  -f $backups/data/backup.tgz
```

Variablen

- In der Bash definierte Variablen . . .
- . . . werden durch `export` zu Umgebungsvariablen.
- Der Inhalt wird mit dem Dollarzeichen angefordert.

Ausnahmen von der “Zeilen”-Regel

- Der Backslash (\ ) fügt Zeilen zu Kommandos zusammen.
- Das Semikolon (;) teilt Zeilen in Kommandos auf.

```
#!/bin/sh
#
# Datei: backup.3
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-25
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

modus="$1"
backups=/home/holger/Vortrag/backups

jaliste="$backups/config/$modus.ja"
neinliste="$backups/config/$modus.nein"
backupdatei="$backups/data/$modus.tgz"

find . -name "*~" -exec rm -f {} ";"

tar -cz \
  --exclude-from=$neinliste --files-from=$jaliste \
  -f $backupdatei
```

Spezielle Variablen

- `$1` bis `$`n sind die Parameter.
- `$@` oder `$*` sind alle Parameter.
- `$#` wird zu der Anzahl der Parameter, die übergeben wurden.
- Leerzeichen trennen Parameter
- Ausnahmen durch Quoting

```
#!/bin/sh
#
# Datei: backup.4
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-08
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

modus="$1"
backups=/home/holger/Vortrag/backups
datum="`date +%Y-%m-%d`"

jaliste="$backups/config/$modus.ja"
neinliste="$backups/config/$modus.nein"
backupdatei="$backups/data/$modus.$datum.tgz"

find . -name "*~" -exec rm -f {} ";"

tar -cz \
  --exclude-from=$neinliste --files-from=$jaliste \
  -f $backupdatei
```

Quoting

- ” (Gänsefüße) Leerzeichen, Hochkommata und “globbing”-Zeichen.
- ’ (Hochkomma) Leerzeichen, Gänsefüße, “globbing”-Zeichen, Dollarzeichen, Backquotes.
- ‘ (Backquotes) Das Ergebnis eines Befehls als Parameter(-liste).

```
#!/bin/sh
#
# Datei: backup.5
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-23
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

modus="$1"
backupdir=/home/holger/Vortrag/backups

jaliste="$backupdir/config/$modus.ja"
neinliste="$backupdir/config/$modus.nein"

if test ! -r "$jaliste"
then
  echo "Datei $jaliste nicht gefunden."
  exit 1
fi

datum="`date +%Y-%m-%d`"
backupdatei="$backupdir/data/$modus.$datum.tgz"
```

```
if test -r "$neinliste"
then
  exclude="--exclude-from=$neinliste"
fi

find . -name "*~" -exec rm -f {} ";"

tar -cz \
  $exclude \
  --files-from $jaliste \
  -f "$backupdatei"

if test "$?" != 0
then
  echo "Problem beim Backup von $modus"
  exit 1
fi

echo "Fertig"
```

Bedingte Anweisungen

- `if` besteht aus drei (vier) Teilen: `if`, `then`, (`else`) und `fi`.
- `test` ist der Standardbefehl.
- `[` ist der `test`-Befehl in Verkleidung.
- Der Exit (Errorlevel) wird als Bedingung benutzt
  (kann mit `$?` abgefragt werden).
- Jeder Befehl kann eine Bedinung darstellen.

```
#!/bin/sh
#
# Datei: backup.6
# Autor: Holger Klawitter <info@klawitter.de>
# Datum: 2001-10-05
#
# Erstellung von Backups andhand von Positiv- und Negativlisten.
#

modus="$1"
backupdir=/home/holger/Vortrag/backups

jaliste="$backupdir/config/$modus.ja"
neinliste="$backupdir/config/$modus.nein"

if [ ! -r "$jaliste" ]; then
  echo "Datei $jaliste nicht gefunden." >&2
  exit 1
fi

datum="`date +%Y-%m-%d`"
backupdatei="$backupdir/data/$modus.$datum.tgz"
```

```
if [ -r "$neinliste" ]; then
  exclude="--exclude-from=$neinliste"
fi

tar -cz \
  $exclude \
  -f "$backupdatei" \
  `grep -v '^#' $jaliste` \
  2>/dev/null

if [ $? != 0 ]; then
  echo "Problem beim Backup von $modus" >&2
  exit 1
else
  echo "Fertig" >&2
fi
```

Verbindung zur Aussenwelt

Rein:

- `stdin` Tastatur oder Eingabedatei
- Parameter "konfigurieren" das Programm.

Raus:

- `stdout` ist für "normale" Ausgabe.
- `stderr` ist für Fehler oder Protokollmeldungen.
- Errorlevel - meldet, ob Kommando erfolgreich war.

Ein- und Ausgabe umleiten

Was	Wie
`stdin` aus Datei lesen	`<Datei`
`stdout` in Datei	`>Datei`
`stdout` an Datei anhängen	`>>Datei`
`stderr` an Datei anhängen	`2>>Datei`
`stderr` in Datei	`2>Datei`
`stdout` nach `stderr`	`>&2`
`stderr` nach `stdout`	`2>&1`
`stdout+stderr` in Datei	`>Datei 2>&1` oder `&>Datei`
`stdout` in Datei und `stderr` nach `stdout`	`2>&1 >Datei`
`stdout` weiterleiten	`prog1 | prog2`

Zusammenfassung

- Die Bash ist auch eine kleine Programmiersprache
- Ein Shellskript ist eine Datei mit einem Befehlsaufruf pro Zeile.
- Man hat lokale und globale Variablen.
- Man kann Parameter durch Quoting zusammenstellen.
- Man kann Programmausgaben mit Umleitung und Pipes verbinden.
- Man kann den Ablauf mit `if`, `for` und `while` steuern.
- Man kann auch Unterprogramme definieren.

Further Reading

- `man bash`
- `www.pl-berichte.de/t_shell/`
- `http://www.klawitter.de/bashvortrag`
- Rathmann, Wieskotten: Jetzt lerne ich Shell-Programmierung, Markt und Technik (DM 50)
- (Das “Krakenbuch”) Helmut Herald: Linux Unix Shells (DM 100)
- Newham / Rosenblatt: Learning the bash shell (DM 100)
- Bash Reference Manual `http://www.gnu.org/manual/bash-2.0.2`
- (Für die vielen Befehle) Günther: Linux Ge-Packt; mitp

Danke für’s Zuhören