Bashscripting 103

Eine Präsentation

von

Marius Schwarz

Bashscripting 103

IF-THEN-ELSE

Anweisungen

Bashscripting 103

Heute geht es um alternative Pfade in Bashscripten.

Ein Bash-Script muß, wie jedes andere Programm, ggf. nur dann etwas tun, wenn bestimmte Bedingungen erfüllt sind.

Beispiel:

„WENN eine Datei nicht gepackt ist, DANN packe sie!“

Bashscripting 103

Schritt 1 – Dateigröße feststellen

Dazu benutzen wir den Befehl *stat*.

```
[marius@eve test]$ stat beispiel.txt
  Datei: beispiel.txt
  Größe: 33585      Blöcke: 72         EA Block: 4096   reguläre Datei
Gerät: fd00h/64768d Inode: 6301963     Verknüpfungen: 1
Zugriff: (0664/-rw-rw-r--)  Uid: ( 1000/  marius)   Gid: ( 1000/  marius)
Kontext: unconfined_u:object_r:user_tmp_t:s0
Zugriff    : 2019-08-07 12:49:09.000000000 +0200
Modifiziert: 2019-08-07 12:49:09.000000000 +0200
Geändert   : 2019-08-07 12:54:00.499665243 +0200
 Geburt    : -
```

Bashscripting 103

Aber wie bekommt man da jetzt die Größe so ausgelesen, daß man damit auch was anfangen kann?

Bashscripting 103

```
BEZEICHNUNG
       stat - Anzeige des Datei- oder Dateisystemstatus

ÜBERSICHT
       stat [OPTION]¼ DATEI¼

BESCHREIBUNG
       Status einer Datei oder eines Dateisystems anzeigen.
...
       -c  --format=FORMAT
              Das  angegebene  FORMAT  anstelle der Vorgabe benutzen; nach jeder
Benutzung von FORMAT einen Zeilenum-
              bruch ausgeben.
¼
       %m     Einhängepunkt
       %n     Dateiname
       %s     Gesamtgröûe in Byte
```

Bashscripting 103

```
[marius@eve test]$ stat -c%s beispiel.txt
33585
```

Das ist natürlich schon sehr viel besser.

Im Bash-Script müssen wir diese Anweisung entweder in Backticks `` setzen (Deprecated), oder (Modern) in $() um die Dateigröße in eine Variable schreiben zu können.

Bashscripting 103

```
#!/bin/bash

FILESIZE=$(stat -c%s "$1")

echo "$1 ist $FILESIZE Bytes groß."
```

Zur Erinnerung:

$1 ist das erste Argument, daß dem Script als Befehl übergeben wird.

z.B. ./test.sh beispiel.txt

Bashscripting 103

Test wir das mal aus:

```
[marius@eve test]$ ./test1.sh beispiel.txt
beispiel.txt ist 33585 Bytes groß.
```

\o/ es klappt \o/

Bashscripting 103

Ob das auch fehlertolerant ist ?

Bashscripting 103

```
[marius@eve test]$ ./test1.sh
stat: der Aufruf von stat für '' ist nicht möglich: Datei oder Verzeichnis nicht gefunden
 ist  Bytes groß.
```

Bashscripting 103

Bashscripting 103

Dann müssen wir das wohl abfangen:

WENN keine Angabe in $1, DANN brich ab.

```
#!/bin/bash

if [ "$1" == "" ]; then
        echo "Sie müssen auch eine Datei angeben!"
        exit;
fi

FILESIZE=$(stat -c%s "$1")

echo "$1 ist $FILESIZE Bytes groß."
```

Bashscripting 103

Jo, geht:

```
[marius@eve test]$ ./test1.sh
Sie müssen auch eine Datei angeben!
[marius@eve test]$
```

Bashscripting 103

Schritt 2 – Dateigröße vergleichen

Dazu benutzen wir die Befehlskette:

if [OPERATION]; then … ANWEISUNG... fi

In einem früheren Beitrag haben wir gelernt, daß „[„ der eigentliche Vergleichsbefehl ist, und if gar kein Linuxbefehl ist, sondern eine Build-In-Function von Bash.

Bashscripting 103

if [OPERATION]; then … ANWEISUNG... fi

If ruft den Linuxbefehle [auf, der führt die VergleichsOPERATION durch und ruft dann in Abhängigkeit des Ausgangs der VergleichsOPERATION die ANWEISUNG aus, was ein Mix aus Linuxbefehlen und Build-In-Functions sein kann.

Beispiel:

```
if [ „A“ == „B“ ]; then echo „A ist gleich B“; fi
```

Bashscripting 103

Zahlen vergleichen

Die Vergleichsoperation besteht aus dem Vergleich bspw. „-gt“ und den Variablen/Abgaben die man vergleichen möchte, hier $A und die Zahl 10.000

Beispiel: if [$A -gt 10000]; then … fi

WENN $A größer ist als „10000“, DANN mach was

Bashscripting 103

mögliche Vergleiche

Zahlen:

-gt größer als
-lt kleiner als
-ge größer oder gleich
-le kleiner oder gleich

Bashscripting 103

mögliche Vergleiche

*Strings:**

== lexikalisch gleich
!= lexikalisch ungleich

Wenn man in der aktiven Sprache sortiert
> kommt der String danach?
< kommt der String davor?

*) Strings sollten immer in Anführungszeichen gesetzt werden: „ABC“ == „$B“

Bashscripting 103

mögliche Vergleiche

Fileoperationen: (hat nur ein rechtes Argument)

-e gibt es die Datei
-f gibt es die Datei und ist das auch eine Datei!
-x ist die Datei ausführbar
-r ist die Datei lesbar

Beispiel: if [-e „/etc/passwd“]; then … fi

Bashscripting 103

```
#!/bin/bash

if [ "$1" == "" ]; then
        echo "Sie müssen auch eine Datei angeben!"
        exit;
fi

FILESIZE=$(stat -c%s "$1")

if [ $FILESIZE -gt 10000 ]; then
        echo "Datei $1 ist größer als 10.000 Bytes"
        # jetzt müssen wir ja noch was machen : bzip2 komprimiert Dateien
        bzip2 -9 $1
fi
```

Bashscripting 103

und funktioniert …

```
[marius@eve test]$ ./test.sh beispiel.txt
Datei beispiel.txt ist größer als 10.000 Bytes
[marius@eve test]$
```

Bashscripting 103

und funktioniert …

```
[marius@eve test]$ ./test.sh beispiel.txt
Datei beispiel.txt ist größer als 10.000 Bytes
[marius@eve test]$
```

Aber was passiert, wenn die Datei schon mit Bzip2 komprimiert wurde?

Bashscripting 103

Bashscripting 103

Schritt 3 – Absicherung gegen doppelte Ausführung

Nicht schwer zu erraten, was passiert, wenn man eine bereits komprimierte Datei nochmals komprimieren möchte: Es geht nicht.

Im Falle von Bzip2 schon, weil das selbst erkennt, daß die Datei bereits ein Bzip2 Komprimat ist. Allerdings kann und darf man sich nicht darauf verlassen, die meisten Befehle machen dann stumpf was sie sollen nochmal :(

Bashscripting 103

Schritt 3 – Absicherung gegen doppelte Ausführung

Also brauchen wir einen Weg um zuerkennen, daß wir das bereits mal mit der Datei gemacht haben.

Bzip2 benennt die Datei nach dem komprimieren in Dateiname.bz2 um.

Dies machen wir uns mit einem Stringvergleich des Dateinamens zu nutze:

Bashscripting 103

```
#!/bin/bash

if [ "$1" == "" ]; then
        echo "Sie müssen auch eine Datei angeben!"
        exit;
fi

FILESIZE=$(stat -c%s "$1")

if [ $FILESIZE -gt 10000 ]; then
        echo "Datei $1 ist größer als 10.000 Bytes"

        if [[ "$1" != *.bz2 ]]; then
                echo "komprimiere Datei $1"
                bzip2 -9 $1
        fi
fi
```

Bashscripting 103

Schritt 3 – Absicherung gegen doppelte Ausführung

Die Vergleichsoperation [["$1" != *.bz2]] zeigt den Einsatz von Wildcards (*).

Dazu darf aber der Prüfstring, hier *.bz2, nicht in Anführungszeichen stehen, da sonst exakt auf "*.bz2" geprüft würde *und nicht, so wie wir das brauchen, auf alles, was .bz2 am Ende hat*.

Es gilt: " … != *.bz2" ungleich " ... != "*.bz2" "

Bashscripting 103

Zusammenfassung

Zusammengefasst erfüllt unser kleines Script folgende Bedingungen:

WENN Dateigröße > 10.000
  UND Dateiname nicht mit „.bz2“ endet,
DANN
  komprimieren die Datei mit Bzip2

Bashscripting 103

Mit ELSE einem alternativen Programmpfad folgen:

```
#!/bin/bash

if [ "$1" == "" ]; then
        echo "Sie müssen auch eine Datei angeben!"
        exit;
fi

FILESIZE=$(stat -c%s "$1")

if [ $FILESIZE -gt 10000 ]; then
        echo "Datei $1 ist größer als 10.000 Bytes"
        if [[ "$1" != *.bz2 ]]; then
                echo "komprimiere Datei $1"
                bzip2 -9 $1
        else
                echo "Datei ist schon bz2 komprimiert"
        fi
else
        echo "Datei $1 ist zu klein, lohnt sich nicht!"

fi
```

Bashscripting 103

Damit sind wir bei IF-THEN-ELSE durch.