Kapitel 6

Shell-Skripte

Übersicht

Bisher haben wir alle verwendeten Befehle und Programme manuell über das Terminal ausgeführt. Wie in den vorhergehenden Kapiteln bereits erwähnt, bietet eine moderne Linux Shell einen Funktionsumfang vergleichbar mit einer Programmiersprache. Komplexe oder wiederkehrende Abläufe können durch Skripte automatisiert werden. Im folgenden Kapitel erhalten Sie Einblick in die Funktionsweise von Shell-Skripten und lernen alle damit verbundenen Grundlagen.

Lernziele
Nach Abschluss des Kapitels

- sind Sie mit den Grundlagen von Shell-Skripten vertraut,
- können Sie Shell-Skripte anpassen und ihre Funktion verstehen,
- können Sie einfache Shell-Skripte erstellen.

6.1 Einleitung

Durch die Eigenschaften eines Mehrbenutzersystems und den Einsatz auf Servern ist die Bedienung über die Shell (Kommandozeile) in unixoiden Betriebssystemen von großer Bedeutung. Die Eingabe von bestimmten Befehlsfolgen in regelmäßigen Abständen ist für die Administration sowie Funktion dieser Systeme oft unerlässlich. Damit die Administratoren/Benutzer nicht stundenlang mit der Eingabe von Kommandos oder Befehlsfolgen beschäftigt sind, können diese Vorgänge durch Shell-Skripte automatisiert bzw. "geskriptet" werden.

6.2 Grundlagen

Durch den großen Funktionsumfang moderner Shells sowie durch die Möglichkeit der individuellen Gestaltung, findet man Shell-Skripte in den unterschiedlichsten Bereichen. Shell-Skripte werden am häufigsten verwendet in:

Ein Shell-Skript kann zwar direkt in der Shell erstellt werden (Standardeingabe in eine Datei Umleiten), aber es empfiehlt sich, dafür einen Texteditor zu verwenden. Es kann sich dabei sowohl um einen grafischen Editor als auch um ein Programm handeln, das direkt im Terminal ausgeführt wird. Abgesehen vom Komfort und der Übersichtlichkeit ist es nicht erforderlich, dass der Editor Schlüsselwörter besonders hervorhebt.

Beispiele für grafische Texteditoren:

Beispiele für Texteditoren für die Konsole:

Beim Erstellen eines Shell-Skriptes sollte man sich vorher Gedanken über den Namen machen. Er sollte auf die Funktion des Skriptes hinweisen. Sinnvollerweise sollte der Name nicht identisch mit einem bereits existierenden Systemprogramm (z.B. ls, mv, cp, ...) sein. Ob ein Name bereits belegt ist, lässt sich überprüfen durch:

```
1
2
```

Mit dem Texteditor nano erstellen wir jetzt unser erstes Shell-Skript (01_firstscript.sh), das ein "Hallo, Welt!äusgibt.

```
1
```

```
1 #!/bin/bash
2 echo
3 echo "Hallo, Welt!"
4 echo
```

Bevor wir unser neu erstelltes Shell-Skript benutzen können, müssen wir es natürlich ausführbar machen. Im Anschluss kann es wie jede andere ausführbare Datei aufgerufen werden.

```
1
2
3
4
5
6
7
```

6.3 Die Syntax von Shell-Skripten

Wie bereits erwähnt, kann man sich mit dem Programm file den Dateityp des erzeugten Shell-Skripts anzeigen lassen.

```
1
```

Man erhält folgende Ausgabe:

```
1 01_firstscript.sh: Bourne-Again shell script, ASCII text executable
```

Linux erkennt, dass es sich bei dieser Datei um ein ausführbares Bash-Skript handelt. Da Linux den Dateityp nicht aufgrund der Dateiendung (.sh) bestimmt, ist diese Information in der Datei enthalten. In unserem Shell-Skript ist dafür die Zeile #!/bin/bash verantwortlich.

```
1
```

```
2
```

Shell-Skripte können für verschiedene Kommandozeileninterpreter geschrieben werden. Es muss immer beachtet werden:

Um unsere Kenntnisse im Bereich Shell-Skripte zu erweitern, erstellen wir ein zweites Skript (02_docommands.sh).

```
1
```

```
1 #!/bin/bash
2 # Ausgabe
3 echo
4 echo "Das zweite Skript gibt die Device-Files aus."
5 echo
6 # ls Ausfuehren
7 ls /dev/
8 # Ausgabe
9 echo
10 echo "Das wars!" # ein weiterer Kommentar
11 echo
```

```
1
2
```

In diesem Shell-Skript wird neben dem echo-Kommando (Teil der Bash-Shell) das Systemprogramm ls aufgerufen, um die Device-Files auszugeben. Wie man sieht, ist der Aufruf eines Programms in einem Skript identisch mit der manuellen Eingabe auf der Kommandozeile. Unser Shell-Skript enthält neben den Befehls- bzw. Programmaufrufen noch Kommentare zum besseren Verständnis. Bei Kommentaren ist folgendes zu beachten:

6.4 Komplexere Shell-Skripte

Shell-Skripte sind vom Funktionsumfang mit einer Programmiersprache vergleichbar. Sie bieten die Möglichkeit der Nutzung von:

Als Beispiel für ein komplexeres Shell-Skript erstellen wir unser drittes Skript (03_helloagain.sh).

```
1
```

```
1 #!/bin/bash
2 echo #empty line
3
4 #use variables
5 gruss="hallo"
6 count=0
7
8 #hello loop
9 while (( $count <= 9 ))
10 do
11         #start of if
12         if (( $count == 0 ))
13         then
14                 echo "Hallo, Welt!"
15         elif (( $count < 9 ))
16         then
17                 echo "Nochmal $gruss!"
18         else
19                 echo "Das letzte Hallo!"
20         fi
21         #end of if
22 ((count = $count + 1))
23
24 done
25 #end of hello loop
26
27 echo #empty line
28 echo "ENDE"
29 echo #empty line
```

Nachdem dieses Shell-Skript erstellt wurde, muss es ebenfalls ausführbar gemacht werden, bevor es gestartet werden kann.

```
1
```

```
2
```

```
3
```

6.5 Weiterführende Informationen

Weiterführende Informationen zum Erstellen von Shell-Skripten können aus zahlreichen Quellen im Internet entnommen werden und sind nicht Teil dieser Vorlesung (zu umfangreich).