Bash Scripting for Beginners Cheat Sheet

by plaguedoctor via cheatography.com/209408/cs/45001/

Skript Datei

Datei anlegen	touch myscript.sh
Datei ausführbar machen	chmod +x myscript.sh
Skript ausführen	sh myscript.sh

Erste Zeile jedes Skripts

#!/bin/bash

Diese Zeile weist das System an, das Skript in der BASH-Shell auszuführen, unabhängig davon, welche Shell der Benutzer standardmäßig verwendet. Das Format ist **#!**, gefolgt vom Pfad zum Interpreter, normalerweise **/bin/bash**.

Input/Output

Mit **Input** sind die Daten oder Informationen gemeint, die ein Programm empfängt oder liest. Die Eingaben kann aus verschiedenen Quellen stammen:
Umgebungsvariablen, Kommandozeilen--Argumente, Dateien
Im Skript wird das **read** Keyword verwendet, um User-Input einzulesen

Unter **Output** versteht man die Informationen oder Daten, die ein Programm generiert oder schreibt. Die Ausgabe kann an verschiedene Ziele geleitet werden:
Umgebungsvariablen, Kommandozeilen--Argumente eines anderen Programms, Dateien
Im Skript wird das **echo** Keyword verwendet, um Output in der Konsole auszugeben

Quoting

'.....'	Variablen werden nicht durch Werte ersetzt
"....."	Variablen werden durch Werte ersetzt
`.....`	Inhalt wird als Kommando ausgeführt
..... <<	Bequem längere Texte ausgeben

Basic Variablen

VAR1=abc

Werden definiert durch Name + Gleichzeichen + Wert (alles ohne Leerzeichen)
Für Strings können Anführungszeichen (einfache oder doppelte benutzt werden, müssen aber nicht)
Zugriff auf den Inhalt der Variablen mit **$VAR1** oder **${VAR1}**

Reservierte Variablen

$$	Prozess-ID
$?	Return-Code
$1 - $n	Übergebene Parameter
$*	Alle Parameter – als ein Wort
$@	Alle Parameter – als List von Wörtern
$0	Skriptname (Aufruf)
$#	Anzahl der Parameter

For Schleife

Einfach:

```
for variable in list; do
    command1
    command2
done
```

Verschachtelt:

```
for i in {1..3}; do
    for j in {a..c}; do
        echo " $i$ j"
    done
done
```

While Schleife

```
while [ condition ]; do
        com mand1
        com mand2
done
```

While-Schleifen führen einen Befehlsblock aus, solange eine angegebene Bedingung erfüllt ist.

Until

```
until [ condition ]; do
        com mand1
        com mand2
done
```

Until-Schleifen ähneln While-Schleifen, werden jedoch ausgeführt, bis eine Bedingung erfüllt ist.

Continue / Break

Auf der anderen Seite wird die **Continue**-Anweisung verwendet, wenn wir eine bestimmte Iteration überspringen müssen. Immer wenn wir eine continue-Anweisung schreiben, wird der gesamte Code nach dieser Anweisung übersprungen und die Schleife wird zur nächsten Iteration fortgesetzt.

Eine **Break**-Anweisung wird verwendet, wenn wir die laufende Schleife beenden möchten, wenn eine bestimmte Bedingung eintritt. Immer wenn eine Break-Anweisung auftritt, wird die Schleife unterbrochen und die Ausführung gestoppt.

If Abfrage

```
if [ condition ]; then
        com mands
elif [ condition ]; then
        com mands
else
        com mands
fi
```

Es gibt mehrere verschieden Schreibweisen und auch Kurzformen

Switch Case

```
case expression in
        pat tern1)
                com mands
               ;;
        pat tern2)
                com mands
               ;;
       *)
```

Switch Case (cont)

```
>       default_commands
     ;;
esac
```

If-Anweisungen werden für komplexe Bedingungen und case-Anweisungen für einfachere, musterbasierte Übereinstimmungen

Funktionen

Funktion ohne Rückgabe:

```
functi on_ name() {
    commands
}
```

Funktion mit Rückgabe::

```
functi on_ name() {
    commands
    return result
}
.
```

Rückgabe: Für Strings muss *echo* verwendet werden und *return* erwartet einen numerischen Return code

Kommandotrenner

kommando1 **;** kommando2	Kommandos werden hintereinander ausgeführt
kommando1 **&** kommando2	Erstes Kommando in Hintergrund
kommando1 **&&** kommando2	Logisch und
kommando1 **||** kommando2	Logisch oder

Kommandosubstitution

1) `CMD_OU T=`ls -la``

2) `CMD_OU T=$(ls -la)`

Bei der Kommandosubstitution wird der Inhalt der Klammern / Backticks als Kommando ausgeführt und das Ergebnis nach „links“ übergeben.
In diesen Beispielen wird die Rückgabe der Kommandos in `$CMD_OUT` gespeichert

Arithmetische Operatoren

-gt	größer als	>
-ge	größer gleich	>=
-lt	kleiner als	<
-le	kleiner gleich	<=
-eg	gleich	==
-ne	nicht gleich	!=

Testen von Dateien

-e	Datei existiert
-f	ist reguläre Datei
-s	nicht größer Null
-d	ist ein Verzeichnis
-L	ist ein symbolischer Link