

Shellskripte

Linux-Kurs der Unix-AG

Zinching Dang

13. Juni 2017



Wiederholung & Vertiefung

Shellskripte

Zusammenfassung & Ausblick

Wiederholung & Vertiefung: Shellbefehle & Links

Hard- und Symlinks

- ▶ Hardlinks zeigen auf die Speicherstelle einer anderen Datei
 - ▶ zwei Dateinamen für ein und die selbe Datei
 - ▶ nur mit Dateien möglich
- ▶ Symlinks zeigen auf eine andere Datei
 - ▶ funktioniert mit Dateien und Verzeichnissen
- ▶ Erkennung von Hard- und Symlinks mit `ls -l`
 - ▶ Hardlink anhand des Link Counts
 - ▶ Symlink anhand dem `l` in der ersten Spalte

Wiederholung & Vertiefung: Shellbefehle & Links

Shellbefehle

- ▶ `ln` bzw. `ln -s` erzeugen Hard- bzw. Symlinks
- ▶ `df` zeigt den Speicherverbrauch an
- ▶ `du` zeigt die Größe von Dateien und Verzeichnissen an
- ▶ `file` zeigt den Dateityp an

Shellskripte

Shellskripte sind

- ▶ eine Aneinanderreihung von Befehlen, die ausgeführt werden
- ▶ durch Kontrollstrukturen steuerbar
- ▶ nützlich bei Routine-Aufgaben, die einem Schema folgen

Shellskripte ausführen

- ▶ müssen mit `chmod +x skript` ausführbar gemacht werden
- ▶ werden mit `./skript` ausgeführt
- ▶ können optional Argumente haben

Shellskripte

Hello World!

```
1 #!/bin/sh  
2 echo "Hello World!"
```

Aufbau

- ▶ fängt mit dem Shebang `#!` an, gefolgt vom Interpreter
- ▶ wird von einem Interpreter ausgeführt
- ▶ hat oft `/bin/sh` oder `/bin/bash` als Interpreter

Shellskripte

Hello World!

```
1 #!/bin/sh
2 echo "Hello World!"
```

Aufbau

- ▶ `echo` gibt Text auf die Standardausgabe aus
- ▶ Text muss in Anführungszeichen stehen
- ▶ Variableninhalte können auch ausgegeben werden

Shellskripte

Kommentare und Einzeiler

- ▶ eine Raute `#` markiert einen Kommentar
- ▶ mehrere Befehle in einer Zeile können mit einem `;` (Semikolon) getrennt werden

Beispiel

```
1 #!/bin/sh
2 echo "Hello World!" ; echo "Hello again!" # ein Kommentar
3 # und noch ein Kommentar
```


Variablen

- ▶ zuweisen mit `Variable=Wert`
 - ▶ Textzuweisung mit `var="bla bla bla"` möglich
- ▶ zugreifen mit `$Variable` oder `${Variable}`
 - ▶ `${Variable}` nützlich, wenn in Zeichenkette eingebettet
- ▶ löschen mit `unset Variable`

Anführungszeichen

- ▶ notwendig wenn Text Leer- oder Sonderzeichen enthält
- ▶ doppeltes Anführungszeichen `"` (<Shift+2>)
 - ▶ ersetzt Variablen durch ihren Wert
- ▶ einfaches Hochkomma `'` (<Shift+#>)
 - ▶ Zeichen werden eins zu eins übernommen
- ▶ Gravis/Backtick ``` (<Shift+'>, rechts neben der ? Taste)
 - ▶ Text wird als Befehl behandelt

Shellskripte

Beispiel zu Anführungszeichen

```
1 #!/bin/sh
2 var="ls"
3 echo "$var" # doppelte Anführungszeichen
4 echo '$var' # Hochkomma
5 echo ` $var ` # Backtick
```

Ausgabe

```
1 ls
2 $var
3 Arbeitsflaeche Bilder Dokument Downloads Vorlagen
```

Argumente

- ▶ Shellskripte können Argumente annehmen
- ▶ werden im Shellskript über Variablen bereitgestellt
 - ▶ `$0` enthält den Skriptnamen
 - ▶ `$#` enthält die Anzahl der Argumente
 - ▶ `$*` enthält alle Argumente
 - ▶ `$1` enthält das erste Argument, `$2` das zweite, ...

Exit-Status

- ▶ Information über (nicht) erfolgreiche Ausführung
 - ▶ 0 steht für erfolgreiche Ausführung
 - ▶ $\neq 0$ steht für nicht erfolgreiche Ausführung
 - ▶ Exit-Status durch Programmierer beliebig festgelegt
- ▶ wird mit `exit 0` gesetzt
- ▶ mit `$?` kann der Exit-Status abgerufen werden

Exit-Status

- ▶ Bedingte Ausführung mit Exit-Status möglich
- ▶ `&&` führt den nachfolgenden Befehl aus, wenn der vorherige Befehl erfolgreich ausgeführt wurde
 - ▶ `rm foo && echo "gelöscht"`
- ▶ `||` führt den nachfolgenden vorherige Befehl aus, wenn der Befehl nicht erfolgreich ausgeführt wurde
 - ▶ `rm foo/ || echo " nicht gelöscht"`

Bedingungen prüfen mit test

Allgemeines

- ▶ `test` prüft eine Bedingung und gibt den Exit-Status 0 zurück, wenn die Bedingung erfüllt ist
- ▶ Kurzschreibweise mit `[]`
 - ▶ Leerzeichen zwischen den Klammern wichtig
- ▶ Bedingung kann mit `!` umgekehrt werden
- ▶ Bedingungen mit Dateien, Zahlen und Zeichenketten möglich

Bedingungen prüfen mit test

Wichtige Bedingungen (Auszug)

- ▶ `-e $datei` wahr, falls `$datei` ein Pfad zu einer Datei ist und existiert (kann auch ein Verzeichnis sein)
- ▶ `-d $verz` wahr, falls `$verz` ein Verzeichnis ist
- ▶ `$n1 -eq $n2` wahr, falls `$n1` und `$n2` gleich groß sind
- ▶ `$n1 -lt $n2` wahr, falls `$n1` kleiner als `$n2` ist
- ▶ `$n1 -gt $n2` wahr, falls `$n1` größer als `$n2` ist
- ▶ `$s1 = $s2` wahr, falls `$s1` und `$s2` übereinstimmen

Kontrollstrukturen

Bedingte Ausführung

```
1 if Bedingung1
2 then
3     Befehl1.1
4     Befehl1.2
5 elif Bedingung2           #optional
6 then
7     Befehl2.1
8     Befehl2.2
9 else                       #optional
10    Befehl3.1
11    Befehl3.2
12 fi
```

Kontrollstrukturen

Beispiel

```
1 #!/bin/sh
2 if [ $# -eq 2 ] # Anz. Parameter
3 then
4     if [ $1 -gt $2 ]; then           # $1 > $2
5         echo $1 "ist_groesser_als" $2
6     elif [ $1 -lt $2 ]; then        # $2 > $1
7         echo $1 "ist_kleiner_als" $2
8     else                             # ansonsten $1 == $2
9         echo $1 "und" $2 "sind_gleich_gross"
10    fi
11 else # nicht genug/zu viele Parameter
12     echo "Es_muessen_2_Parameter_uebergeben_werden!"
13 fi
```

Wiederholte Ausführung

```
1 for i in liste
2 do
3     Befehl1
4     Befehl2
5 done
```

- ▶ `i` ist die Laufvariable
- ▶ `liste` kann ein Array sein
 - ▶ kann mit `liste='1 2 3'` erzeugt werden (einfaches Hochkomma)
 - ▶ kann mit `liste='seq 1 3'` Ausgabe eines Befehls sein

Kontrollstrukturen

Beispiel

```
1 #!/bin/sh
2 liste='1 2 3'
3 for i in $liste; do
4     echo $i
5 done
```

```
1 #!/bin/sh
2 for i in `seq 1 3`; do
3     echo $i
4 done
```

Weitere Shellskript-Features

Interaktion

- ▶ `read` liest von der Standardeingabe
 - ▶ `read bla` speichert die Eingabe in der Variablen `bla`
- ▶ `sleep 5` wartet 5 Sekunden und tut nichts

Rechnen

- ▶ `$((Ausdruck))` wertet den Ausdruck in der doppelten Klammer arithmetisch aus
 - ▶ `x=$((x+1))` inkrementiert die Variable `x` (Voraus. `x` ist eine Zahl)

Zusammenfassung & Ausblick

Zusammenfassung

- ▶ Aufbau von Shellskripten
- ▶ Verwendung von Kontrollstrukturen

Nächstes Mal

- ▶ Benutzer- und Rechteverwaltung
- ▶ Systemverwaltung
- ▶ Netzwerkgrundlagen