Betriebssysteme

Übungen 04

Prof. Dr. Rainer Werthebach

Studiengang Informatik

Hochschule Aalen - Technik und Wirtschaft

case-Anweisung

- Allgemein:

```
case $variable in
  muster 1 ) kommandoliste ;;
  muster 2 ) kommandoliste ;;
  …
  muster n ) kommandoliste n ;;
esac
```

- Beispiele für Muster:

1)	→ der Wert 1
[abc])	→ a, b oder c
*.doc)	→ Endung .doc
*)	→ Default
*)	→ der Wert *

case-Anweisung Beispiel

```
#!/bin/bash
echo -n "Wollen Sie weitermachen? (J/N)? "
read x
case $x in
    [Jj]*) echo "Ja" ;;
    [Nn]*) echo "Nein" ;;
esac
```

test-Kommando

- `test` *`Ausdruck`* → Testet den Wahrheitswert des Ausdrucks

  wird meistens im Zusammenhang mit der if-Anweisung genutzt

  alternative Schreibweise:

  `[␣Ausdruck␣]`

- Wichtige Ausdrücke:

[-d name]	→	Verzeichnis existiert
[-f name]	→	Datei existiert
[-s name]	→	Datei existiert und ist nicht leer
[-r name]	→	Datei existiert und ist lesbar
[-w name]	→	Datei existiert und ist beschreibbar
[-x name]	→	Datei existiert und ist ausführbar
[string]	→	String ist nicht der Nullstring
[name1 = name2]	→	Strings sind identisch
[zahl1 -eq zahl2]	→	Zahlen sind gleich

test-Komando

- Verknüpfungen:
  - -a → and
  - -o → or
  - ! → not

- Beispiele:

```
[ -d name1 -a -f name2 ]
[ ! -r name ]
```

if-Anweisung

- Syntax:

```
if Bedingung
then
   Kommandoliste
elif Bedingung
then
   Kommandoliste
else
   Kommandoliste
fi
```

if-Anweisung - Beispiel

```
#!/bin/bash
for Dateiname # entspricht: for Dateiname in $@
do
    if [ -d $Dateiname ]
    then
            echo "$Dateiname ist ein Verzeichnis!"
    elif [ ! -s $Dateiname ]
    then
            echo "$Dateiname ist leer!"
    else
            echo "$Dateiname wird ausgegeben"
            cat $Dateiname
    fi
done
```

while-Schleife

- Syntax:

```
while Bedingung
do
   Kommandoliste
done
```

- Bedeutung:

  Solange Bedingung wahr ist, wird die Kommandoliste ausführt.

until-Schleife

- Syntax:

```
until Bedingung
do
   Kommandoliste
done
```

- Bedeutung:

  Bis die Bedingung wahr ist, wird die Kommandoliste ausführt.

Schleifen - Beispiel

```
#!/bin/bash
while [ $1 ]
do
    Anzahl=$(cat $1 | wc -w)
    echo "Die Anzahl Woerter in $1 ist $Anzahl"
    shift 1
done
```

Funktionen

- Syntax:

```
function Funktionsname {
   Kommandos
}
Funktionsname () {
   Kommandos
}
```

- Bash-Funktionen liegen im Hauptspeicher (schneller Zugriff).
- Bash-Prozeduren (externe Skripte) starten langsamer, da sie immer neu geladen werden.
- Parameter einer Funktion werden wie positionale Skript-Parameter behandelt.

Funktionen (Fortsetzung)

- Funktionsaufruf:

  `Funktionsname [Parameter1 Parameter2 …]`

  Funktion muss vor dem Aufruf deklariert werden

- Funktionsrückgabe:

  `return n` (n = Integer)

  Alternativ über globale Variablen

- Funktionsrückgabe Auslesen:

  `$?` auslesen

  Alternativ: globale Variable auslesen

Steuerkommandos

- `exit [Status]` → beendet Shell mit `Status` oder falls `Status` nicht gegeben mit exit-Status des letzten Kommandos
- `break` → beendet aktuelle Schleife und setzt nach `done` fort
- `continue` → bricht den aktuellen Iterationsschritt ab und macht mit dem nächsten weiter
- `true` → immer wahr
- `false` → immer falsch

Unbedingte / Bedingte Ausführung

- Unbedingte Ausführung:

  ```
  Kommando; Kommando; Kommando; …
  ```

  Beispiel:

  ```
  cat datei1.txt; wc -w datei2.txt
  ```

- Kurzschreibweise für bedingte Ausführungen:

  Variante 1: **IF** Bedingung **THEN** Kommando
  Bedingung **&&** Kommando

  Beispiele:
  ```
  -f datei ] && cat datei
  chmod u+x skript.sh && ./skript.sh
  ```

  Variante 2: **IF NOT** Bedingung **THEN** Kommando
  Bedingung **||** Kommando

  Beispiel:
  ```
  cat datei1.txt || cat datei2.txt
  ```

Bedingte Ausführung - Beispiel

```
#!/bin/bash
nummer=$#
if [ $nummer -eq 0 ]
then
    while true
    do
        read in || break
        echo $in >> temp$$
    done
    nummer=$(cat temp$$ | wc -w)
    rm temp$$
fi
echo "Die Anzahl Wörter ist $nummer."
```