9.Übung

Systemsoftware (SYS)

Christian Baun
cray@unix-ag.uni-kl.de

Hochschule Mannheim – Fakultät für Informatik
Institut für Robotik

30.11.2007

Wiederholung vom letzten Mal

- Die Shell
- Varianten der Shell
- Kommentare
- Auswahl der Shell
- Shell-Skripte testen (`sh`)
- Feste Variablen (`${var}`, `$0`, `$# $*`, `$@`, `$$`, `$-`, `$_`, `$?`, `&!`)
- Kommandozeilenparameter verarbeiten
- Tests für Zeichenketten, Zahlen und Dateien (`test`, `[`)
- Rückgabewert setzen (`true`, `false`)
- Shell-Skripte vorzeitig beenden (`exit`)

Heute

- Shell-Skripting (Teil 2)
  - Kontrollstrukturen in Shell-Skripten
  - `if`-Anweisung
  - `case`-Anweisung
  - `while`-Schleife
  - `until`-Schleife
  - `for`-Schleife
  - Schleifen vorzeitig verlassen (`break`)
  - Schleifen erneut durchlaufen (`continue`)
  - Endlosschleifen

Kontrollstrukturen in Shell-Skripten

- Für Shell-Skripte stehen verschiedene Kontrollstrukturen zur Verfügung.
- **Bedingte Programmausführung**
  - `if`
  - `case`
- **Schleifen**
  - `while`
  - `until`
  - `for`

Bedingte Ausführung mit `if`

- Mit der `if`-Anweisung ist es möglich, Bedingungen zu realisieren.
- Die Struktur der `if`-Anweisung ist:

```
if [ Bedingung ]
then
  Anweisungsblock
fi
```

- Die Bedingung entspricht der Schreibweise von `test`.
- Das `fi` bedeutet *end if*.

Beispiel zur `if`-Anweisung

```
# cat if
#!/bin/bash
# Beispiel zur if-Anweisung

if [ `whoami` == "root" ]
then
  echo "Sie sind der Admin."
fi
```

```
# ./if
Sie sind der Admin.
```

Bedingte Ausführung mit `if-else`

- Die `if`-Anweisung kann um einen `else`-Zweig erweitert werden.
- Die Struktur der `if-else`-Anweisung ist:

```
if [ Bedingung ]
then
  Anweisungsblock
else
  Anweisungsblock
fi
```

Beispiel zur `if-else`-Anweisung

```
$ cat ifelse
#!/bin/bash
# Beispiel zur if-else-Anweisung

if [ `whoami` == "root" ]
then
  echo "Sie sind der Admin."
else
  echo "Sie sind nicht der Admin."
fi
```

```
$ ./ifelse
Sie sind nicht der Admin.
```

Bedingte Ausführung mit `if-elif-else`

- Die `if-else`-Anweisung kann um einen oder mehr `elif`-Zweige erweitert werden.
- Die Struktur der `if-elif-else`-Anweisung ist:

```
if [ Bedingung ]
then
  Anweisungsblock
elif [ Bedingung ]
then
  Anweisungsblock
else
  Anweisungsblock
fi
```

Beispiel zur `if-elif-else`-Anweisung

```
$ cat ifelifelse
#!/bin/bash
# Beispiel zur if-elif-else-Anweisung

if [ `whoami` == "root" ]
then
  echo "Sie sind der Admin."
elif [ `whoami` == "alice" ]
then
  echo "Sie sind Alice."
elif [ `whoami` == "bob" ]
then
  echo "Sie sind Bob."
else
  echo "Keine Ahnung, wer Sie sind."
fi
```

Weiteres Beispiel zur `if-elif-else`-Anweisung

```
$ cat umfrage
#!/bin/bash
# Beispiel zur if-elif-else-Anweisung

echo "Finden Sie Shell-Skripting schwer? (ja/nein)"
read antwort
echo "Ihre Antwort war: $antwort"
if [ "$antwort" = "ja" ]
then
  echo "üben üben üben."
elif [ "$antwort" = "nein" ]
then
  echo "weiter so."
else
  echo "Diese Antwort habe ich nicht verstanden."
fi
```

Die `case`-Anweisung

- Die `case`-Anweisung wertet einen einzelnen Wert aus und verzweigt zu einem passenden Code-Abschnitt.
- Die Struktur der `case`-Anweisung ist:

```
case Variable in
  Muster) Anweisungsblock ;;
  Muster) Anweisungsblock ;;
  Muster) Anweisungsblock ;;
  *) Default-Anweisungsblock ;;
esac
```

- Die Anweisungsblöcke werden durch `;;` abgeschlossen, denn ein einzelnes Semikolon ist das Trennzeichen zwischen Kommandos auf der selben Zeile.
- Das `esac` bedeutet *end case*.

Beispiel zur `case`-Anweisung

```
#!/bin/bash
# Beispiel zur case-Anweisung

echo "Finden Sie Shell-Skripting schwer?"
read antwort
case "$antwort" in
  j*|J*|y*|Y*)
    echo "üben üben üben."
  ;;
  n*|N*)
    echo "weiter so."
  ;;
  *)
    echo "Diese Antwort habe ich nicht verstanden."
  ;;
esac
```

Die `while`-Schleife

- Eine `while`-Schleife wiederholt eine Menge von Befehlen, **so lange eine Bedingung erfüllt ist**.
- Bei `while` erfolgt die Prüfung der Bedingung **vor** der Abarbeitung der Schleife.
- Die Struktur der `while`-Schleife ist:

```
while [ Bedingung ]
do
  Anweisungsblock
done
```

Beispiel zur `while`-Schleife

```
#!/bin/bash
# while-Schleife

i=1

while [ $i -le 5 ]
do
  echo $i
  i=‘expr $i + 1‘
done
```

```
$ ./while
1
2
3
4
5
```

- Achtung: Die Hochkommata, die `expr` umschließen, sind diejenigen, neben der Backspace-Taste (Shift nicht vergessen!)

Die `until`-Schleife

- Eine `until`-Schleife wiederholt eine Menge von Befehlen, **so lange bis eine Bedingung erfüllt ist**.
- Bei `until` erfolgt die Prüfung der Bedingung **nach** der Abarbeitung der Schleife.
- Die Struktur der `until`-Schleife ist:

```
until [ Bedingung ]
do
  Anweisungsblock
done
```

Beispiel zur `until`-Schleife

```
#!/bin/bash
# until-Schleife

i=1

until [ $i -gt 5 ]
do
  echo $i
  i=‘expr $i + 1‘
done
```

```
$ ./until
1
2
3
4
5
```

Die `for`-Schleife

- Die `for`-Schleife iteriert über Werte aus einer Liste.
- Die Struktur der `for`-Schleife ist:

```
for variable in Liste_der_Parameter
do
  Anweisungsblock
done
```

Beispiel zur `for`-Schleife

```
$ cat for
#!/bin/bash
# for-Schleife

for nummer in eins zwei drei
do
  echo "Parameter $nummer"
done
```

```
$ ./for
Parameter eins
Parameter zwei
Parameter drei
```

`break` **und** `continue`

- Um eine Schleife vorzeitig zu verlassen, existiert das Kommando `break`.
- Ein Aufruf von `break` weist die Shell an, zur nächsten Anweisung hinter der Schleife zu springen.
- Das Gegenstück von `break` ist `continue`.
- Ein Aufruf von `continue` weist die Shell an, zum Anfang der Schleife zurückzukehren und gegebenenfalls einen neuen Durchlauf zu starten.
- Sind mehrere Schleifen ineinander verschachtelt, kann mit einem zusätzlichen Argument ausgewählt werden, welche Schleifen die Shell abbrechen bzw. wiederholen soll.
  - `break` oder `break 1` beendet die direkt umgebende Schleife.
  - `break 2` beendet die zweite von innen umgebende Schleife usw.
  - `continue 2` beendet die innere Schleife und startet die äußere neu.

Einsatzbeispiele von `break`

- `break` und `sleep` in Warteschleifen

```
while true
do
  [ -f datei.tmp ] && break
  sleep 60
done
```

- `break` bei Benutzereingaben

```
while true
do
  read eingabe
  [ $eingabe == "q" ] && break
  ...
done
```

Endlosschleifen

- Endlosschleifen können auf zwei Arten einfach realisiert werden:
  - Mit einer `while`-Schleife und der Bedingung `true`.
  - Mit einer `until`-Schleife und der Bedingung `false`.

```
$ cat while_endlos
#!/bin/bash

while true
do
  echo "Endlosschleife"
done
```

```
$ cat until_endlos
#!/bin/bash

until false
do
  echo "Endlosschleife"
done
```

- Solche Schleifen werden in der Regel auf Grund einer Bedingung mit `break` oder `exit` beendet.

Endlosschleifen mit `break` beenden

```
while true
do
  Kommandos
  ...
  if [ Bedingung ]
    then break
  fi
  ...
done
```

- Oder einfacher:

```
  ...
  if [ Bedingung ] && break
```

Nächste Übung:

7.12.2007