Korn-Shell: Einführung in Shellscripte

Korn-Shell: Einführung in Shellscripte

- Schnittstelle zwischen Nutzer und Betriebssystem

- Unterschieden wird zwischen Grafischen Benutzeroberflächen(GUI) und Zeichen Orientierten Benutzerschnittstelle (TUI)

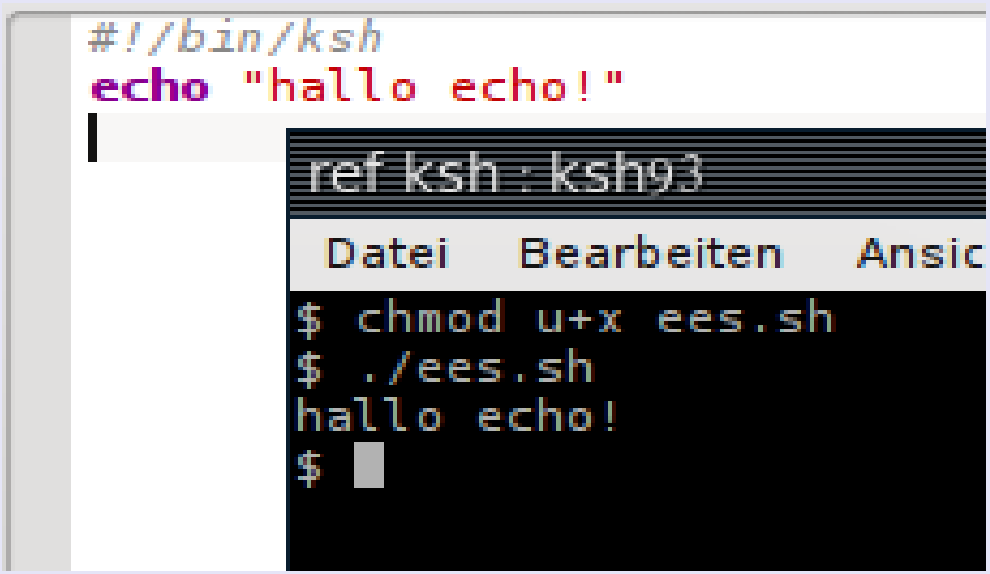
- Shell als Commandline Interpreter der ein Commandline Interface zur Verfügung stellt

Korn-Shell: Einführung in Shellscripte

- Seit 1982 für Unix System V verfügbar
- direkte Weiterentwicklung der Burne Shell
- unter CPL frei Verfügbar und für viele Systeme portiert
- Shell Language Standard (IEEE Posix 1003.2) Konform

Korn-Shell: Einführung in Shellscripte

- Handwerkszeug sind ein Texteditor und die korn-Shell selbst



- *shebang* - Zeile zur Angabe des intepretierenden Programmes
- Ausführungsberechtigung muss erteilt werden
- Start mit ./scriptname.sh

Korn-Shell: Einführung in Shellscripte

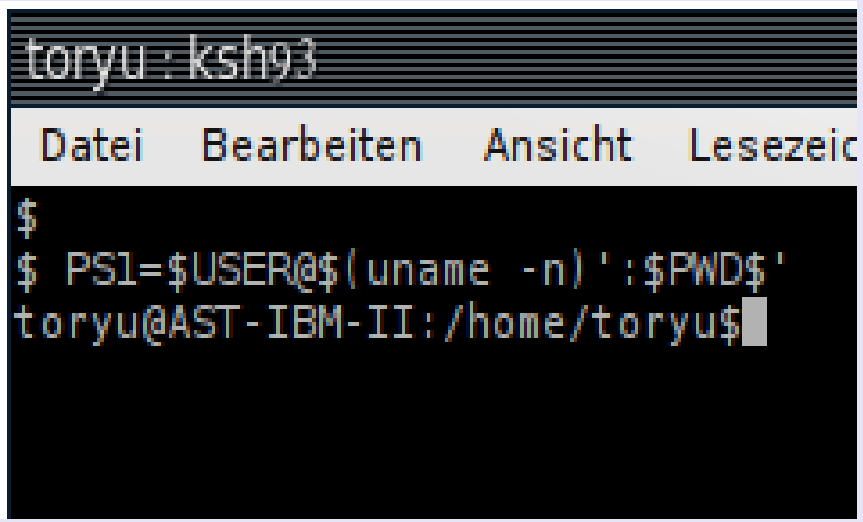
- vereinfachen und automatisieren Aufgaben
- Verbinden bestehende Komponenten um Anwendungen zu erstellen
- Stellen einfach Kontroll- und Datenspeicherstrukturen zur Verfügung

Korn-Shell: Einführung in Shellscripte

- dient zur Statusausgabe
- ksh zeigt beim Start nur als was wir Eingeloggt sind:

  # = root

  $ = Standard-User



- Prompt Definition erfolgt über die Variablen PS1 bis PS4
- Bsp: *PS1=$USER@$(uname -n)':$PWD: ' export PS1*

Korn-Shell: Einführung in Shellscripte

- Wiederbenutzung von Befehlen
- *history*-Befehl zeigt nummerierte Liste der letzten genutzten Befehle
- Mit *r befehlsnummer* wird der gewünschte Befehl erneut ausgeführt
- Befehlsspeicher ist begrenzt und kann mit `HISTSIZE=n; export HISTSIZE` verändert werden

Korn-Shell: Einführung in Shellscripte

- Variablen haben einen Datentyp(int, string, etc.)
- ksh stellt verschiedene Systemvariablen zur Verfügung ($0..{10}, PS1..4) die über den Befehl ***set*** eingesehen werden können
- Initialisierung über ***my_var=ein_wert***
- Benutzung des Inhalts mit ***$my_var***
- Zurücksetzen mit ***unset my_var***

Korn-Shell: Einführung in Shellscripte

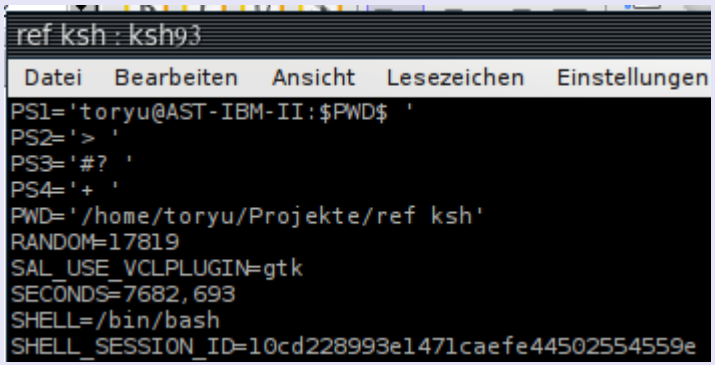
- Datentypen Bestimmen den Inhalt einer Variable
- Ksh unterscheidet u. A.:
  integer Variablen (-i), exportierte Variablen (-/+x), readonly Variablen (-r)
- Deklaration über **typeset**
  Bsp: *typeset -r -Z -i n=2*

```
#!/bin/ksh
typeset i;
n1=12;
n2=12;
let e1=n1+n2;
echo $e1;
```

```
ref ksh : ksh93
Datei  Bearbeiten  Ansicht  Lesezeichen  Einstellungen  Hilfe
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$ ./ees.sh
24
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$
```

Korn-Shell: Einführung in Shellscripte

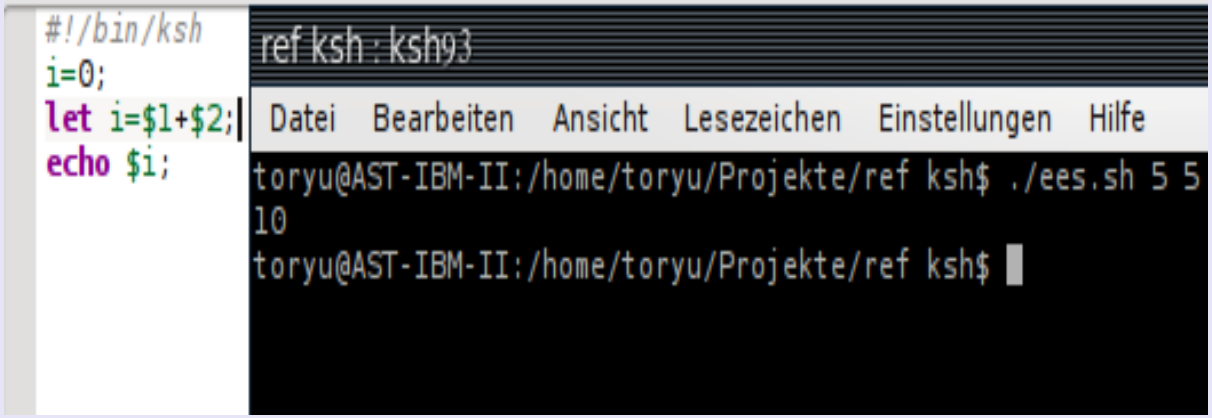
- Umgebungsvariablen müssen weder initialisiert noch deklariert werden, ihre Inhalte können aber Überschrieben werden
- Beispiele sind u. A .: *RANDOM, LANG, PS1*
- Viele Umgebungs-variablen sind importiert



Korn-Shell: Einführung in Shellscripte

- Parameter werden beim Skriptstart übergeben und übernehmen wichtige Steuer- und Wertübergabefunktionen
- Bsp: *./myscript.sh -v*

```
#!/bin/ksh
i=0;
let i=$1+$2;
echo $i;
```



- Die Parameter werden als $0 bis ${nn} übernommen

Korn-Shell: Einführung in Shellscripte

- Grundlage der Interaktion zwischen Programm und Benutzer
- einfache Ausgabe über *echo*
- ergänzend *print* und *printf* vorhanden
- Eingabe über *read*
- Standardmäßig erfolgt die Ausgabe auf den Bildschirm, kann aber auch umgelenkt werden

Korn-Shell: Einführung in Shellscripte

- Generell werden mathematische, vergleichende, zuweisende und logische Operatoren
- logische und verknüpfende Operatoren sind die Grundlage von Steuer und Kontrollstrukturen

```
#!/bin/ksh
if (($1==5));
then
echo "true";
else
echo "false"
fi
```

```
ref ksh : kshg3
Datei  Bearbeiten  Ansicht  Lesezeichen  Einstellungen  Hilfe
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$ ./ees.sh 5
true
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$
```

- Über *expr* und *let* lässt sich der Operator Syntax Steuern

Korn-Shell: Einführung in Shellscripte

- dienen zum erreichen Verschiedener Programmteile bei Unterschiedlichen Bedingungen

- Fallunterscheidung über if … then ... elsif … then ... else ... oder case-Konstrukt

```
1  #!/bin/ksh
2  echo yes or no?
3  read answer
4  case $answer in
5    yes|Yes|y)
6      echo "yes!"
7      ;;
8    no|n)
9      echo "oh no!"
10     ;;
11   q|Q)
12     echo "quit!"
13     exit
14     ;;
15   *)
16     echo "maybe."
17     ;;
18 esac
19
```

```
ref ksh : ksh93
Datei   Bearbeiten
$ ./ees.sh
yes or no?
y
yes!
$ r
./ees.sh
yes or no?
n
oh no!
$ r
./ees.sh
yes or no?
blub
maybe.
$
```

Korn-Shell: Einführung in Shellscripte

- Unterscheidung von Kopf gesteuerten und Fuß gesteuerten Schleifen
- dienen zum wiederholten Ausführen von Befehlen bis zu einer Abbruchbedingung
- ksh bietet *while*, *until* und *for*-Schleifen

Korn-Shell: Einführung in Shellscripte

- Syntaxbeispiel while-Schleife

```
while test –n „$1“
do
   case $1 in
   –*) echo „option: $1“;;
    *) echo „Argumente“
   esac
shift
done
```

Korn-Shell: Einführung in Shellscripte

- Syntaxbeispiel until-Schleife

```
until [[ $answer = „yes“ ]];do
  print –n „please enter \“yes\“: „
  read answer
  print „“
done
```

Korn-Shell: Einführung in Shellscripte

- Syntaxbeispiel for-Schleife

```
for var in $(ls);do
  if [[ -d $var ]];then
    echo $var is a directory
  else
    print $var is not a directory
  fi
done
```

Korn-Shell: Einführung in Shellscripte

Vielen Dank für die Aufmerksamkeit...