

Korn-Shell: Einführung in Shellskripte

1. Übersicht: Einführung 2. Die Kornshell im Detail 3. Grundlagen der Programmierung

1. Übersicht und Einführung

- 1.1 Die Shell allgemein
- 1.2 Die korn-Shell
- 1.3 Der Weg zum ersten Skript
- 1.4 Nutzen und Grenzen von Skripten

2. Die korn-Shell im Detail

- 2.1 Prompts
- 2.2 Variablen und Datentypen
- 2.3 Umgebungsvariablen
- 2.4 Parameter

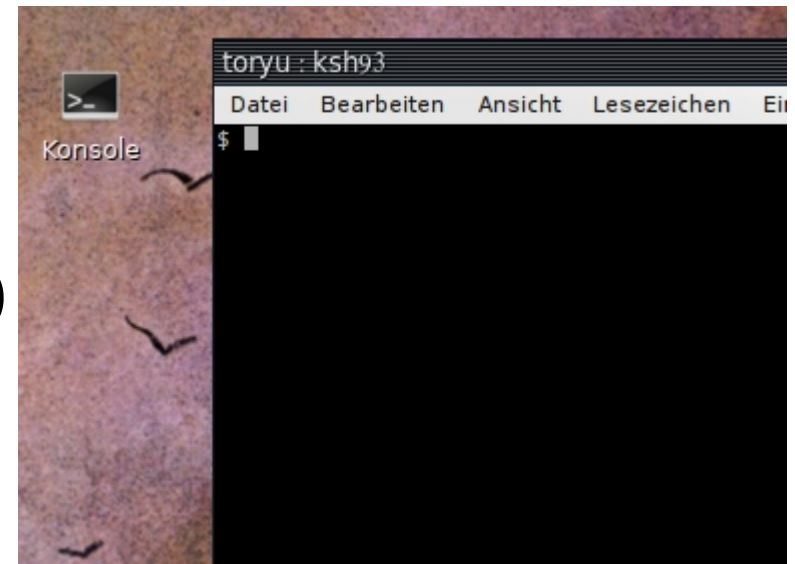
3. Grundlagen der Programmierung

- 3.1 Einfache Ein- und Ausgabe
- 3.2 Steuerstrukturen und logische Operatoren
- 3.3 Bedingte Anweisungen
- 3.4 Schleifen

Korn-Shell: Einführung in Shellskripte

1. Übersicht: Die korn-Shell 2. Die Kornshell im Detail 3. Grundlagen der Programmierung

- 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 Shellskripte

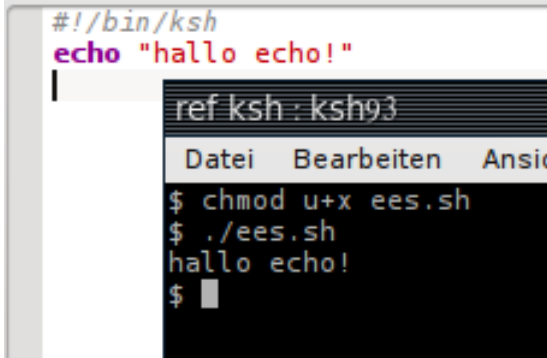
1. Übersicht: Die Korn-Shell 2. Die Kornshell im Detail 3. Grundlagen der Programmierung

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

Korn-Shell: Einführung in Shellscripte

1. Übersicht: Der Weg zum ersten Skript 2. Die Kornshell im Detail 3. Grundlagen der Programmierung

- Handwerkszeug sind ein Texteditor und die Korn-Shell selbst
- shebang - Zeile zur Angabe des interpretierenden Programmes
- Ausführberechtigung muss erteilt werden
- Start mit ./scriptname.sh



```
#!/bin/ksh
echo "hallo echo!"

ref ksh : ksh93
Datei Bearbeiten Ansicht
$ chmod u+x ees.sh
$ ./ees.sh
hallo echo!
$
```

Korn-Shell: Einführung in Shellskripte

1. Übersicht: Nutzen und Grenzen 2. Die Kornshell im Detail 3. Grundlagen der Programmierung

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

Korn-Shell: Einführung in Shellscripte

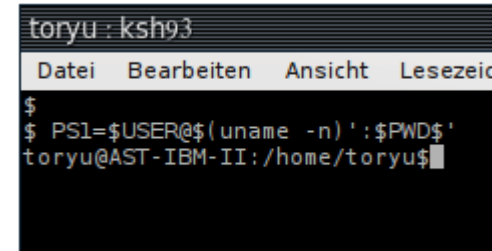
1. Übersicht: 2. Die Kornshell im Detail: Prompts 3. Grundlagen der Programmierung

- 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*



```
toryu : ksh93
Datei  Bearbeiten  Ansicht  Lesezeic
$
$ PS1=$USER@$$(uname -n)':$PWD$ '
toryu@AST-IBM-II:/home/toryu$
```

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshell im Detail: Variablen 3. Grundlagen der Programmierung

- 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***

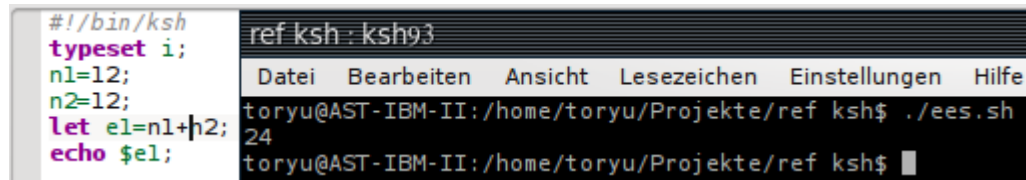
Korn-Shell: Einführung in Shellscripte

1. Übersicht 2. Die Kornshell im Detail: Datentypen 3. Grundlagen der Programmierung

- 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`

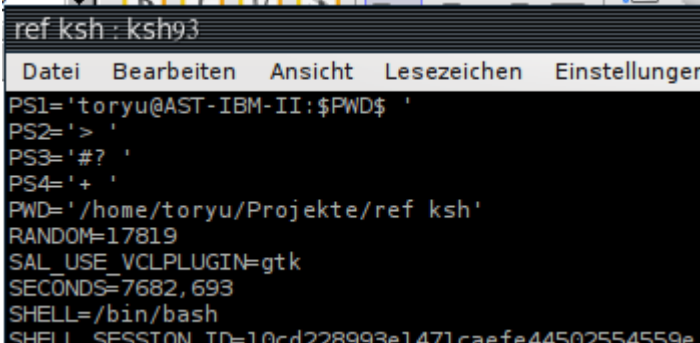


The screenshot shows a terminal window with a menu bar at the top containing 'Datei', 'Bearbeiten', 'Ansicht', 'Lesezeichen', 'Einstellungen', and 'Hilfe'. The title bar reads 'ref ksh : ksh93'. The terminal content shows a script being executed: `#!/bin/ksh`, `typeset i;`, `n1=12;`, `n2=12;`, `let e1=n1+h2;`, and `echo $e1;`. The prompt is `toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$`. The command `./ees.sh` has been entered, and the output `24` is displayed on the next line.

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshell im Detail: Datentypen 3. Grundlagen der Programmierung

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



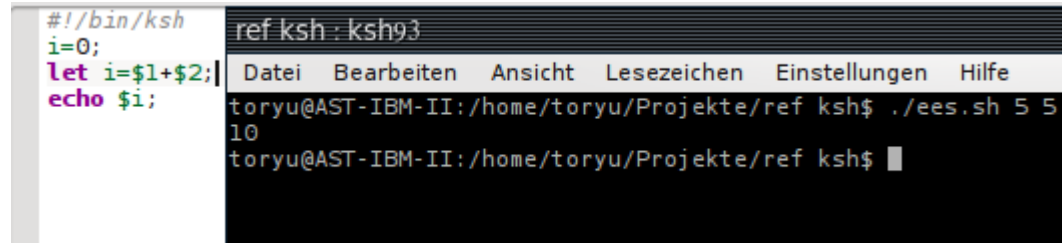
```
ref ksh : ksh93
Datei  Bearbeiten  Ansicht  Lesezeichen  Einstellungen
PS1='toryu@AST-IBM-II:$PWD$ '
PS2='> '
PS3='#? '
PS4='+ '
PWD='/home/toryu/Projekte/ref ksh'
RANDOM=17819
SAL_USE_VCLPLUGIN=gtk
SECONDS=7682.693
SHELL=/bin/bash
SHELL_SESSION_ID=10cd228993e1471caefe44502554559e
```

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshell im Detail: Parameter 3. Grundlagen der Programmierung

- 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;
```

```
ref ksh : ksh93
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$ ./ees.sh 5 5
10
toryu@AST-IBM-II:/home/toryu/Projekte/ref ksh$
```

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

Korn-Shell: Einführung in Shellskripte

1. Übersicht

2. Die Kornshell im Detail: Parameter

3. Grundlagen der Programmierung

- Grundlage der Interaktion zwischen Programm und Benutzer
- einfache Ausgabe über `echo`
- ergänzend *print* und *printf* vorhanden
- Eingabe über *read*

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshel 3. Grundlagen der Programmierung: Operatoren

- Generell werden mathematische, vergleichende, zuweisende und logische Operatoren
- logische und verknüpfende Operatoren sind die Grundlage von Steuer und Kontrollstrukturen
- Über *expr* und *let* lässt sich der Operator Syntax Steuern

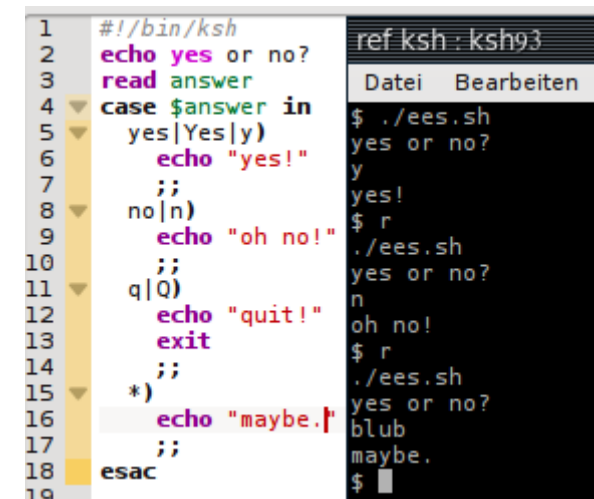
```
#!/bin/ksh
if (($1==5));
then
echo "true";
else
echo "false"
fi
```

```
ref ksh : ksh93
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$ █
```

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die korn-Shel 3. Grundlagen der Programmierung: Bedingte Anweisungen

- dienen zum Erreichen verschiedener Programmteile bei unterschiedlichen Bedingungen
- Fallunterscheidung über
if ... then ... elif ... 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
```

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 Shellskripte

1. Übersicht 2. Die Kornshel 3. Grundlagen der Programmierung: Schleifen

- 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 Shellskripte

1. Übersicht 2. Die Kornshel 3. Grundlagen der Programmierung: While

- Syntaxbeispiel while-Schleife

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

Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshel 3. Grundlagen der Programmierung: While

- Syntaxbeispiel until-Schleife

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


Korn-Shell: Einführung in Shellskripte

1. Übersicht 2. Die Kornshel 3. Grundlagen der Programmierung: While

- 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
```