

Skriptprogrammierung Teil I

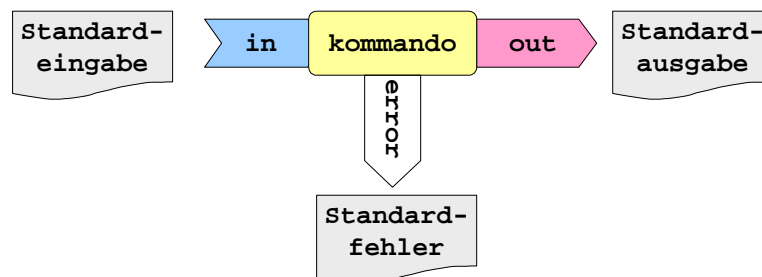
Themen

- ▶ Verwendete Shell ist grundsätzlich die **bash**
Andere Shells können sich teilweise gravierend unterscheiden!
- ▶ Themen
 - Ein- und Ausgabeumleitungen, Pipe
 - Werkzeuge
 - Variable
 - Arithmetik
 - Bedingungen

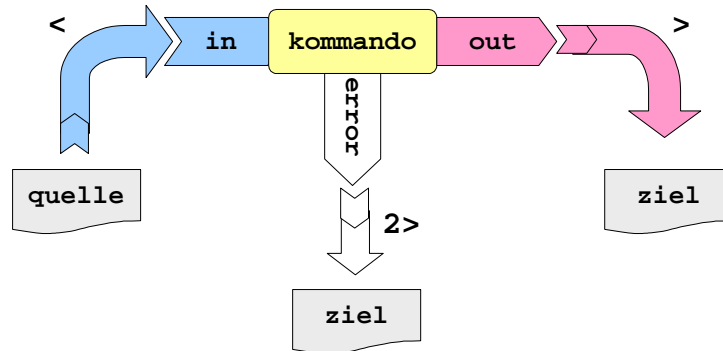
Ressourcen

- ▶ Newham, C.; Rosenblatt, B.:
Lerning the bash Shell
O'Reilly & Associates, USA ²1998
ISBN 1-56592-347-2 (\$ 24.95)
- ▶ Krienke, R.:
UNIX Shell-Programmierung
Carl Hanser Verlag, M"unchen ²2001
ISBN 3-445-21722-3 (DM 39,80)

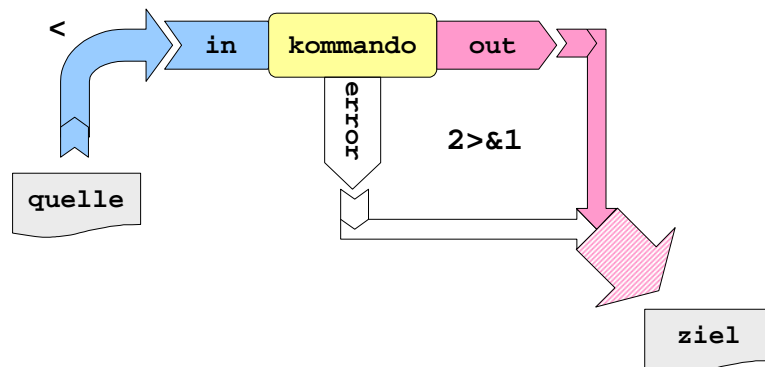
Kommando



Umleitungen

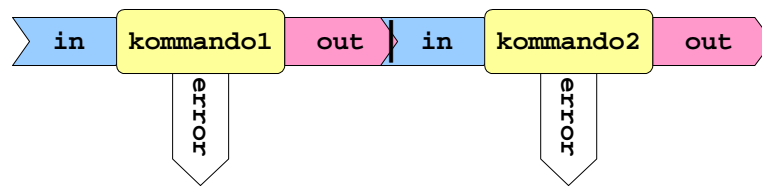


Umleitungen



Umleitungen

- ▶ Oft benötigt:
Ausgabe eines Kommandos wird
Eingabe eines zweiten Kommandos
- ▶ Konzept: `pipe` |



- ▶ Beispiel: `ls -l | more`

wichtige Werkzeuge

- ▶ `cat`
- ▶ `echo`
- ▶ `grep` (get regular expression)
- ▶ `cut` Spalten ausschneiden
 - Spaltentrennzeichen: `-d"<zeichen>"`
 - Spalten auswählen: `-f<bereich>`
- ▶ `sort` sortieren
- ▶ `uniq` doppelte Zeilen verwerfen
- ▶ `sleep` nichts tun

Kommandoverknüpfung

- ▶ `kommando1 ; kommando2`
 - `kommando2` nach `kommando1`
- ▶ `{ kommando1 ; kommando2 }`
 - erscheint als *ein* Kommando
- ▶ `kommando1 && kommando2`
 - `kommando2` nur wenn `kommando1` fehlerfrei lief
- ▶ `kommando1 || kommando2`
 - `kommando2` nur wenn `kommando1` nicht fehlerfrei lief

Variable

- ▶ Kennzeichnung durch führendes `$`
- ▶ vorbelegte Namen (Auswahl):
 - `$HOME` Heimatverzeichnis
 - `$PWD` aktuelles Verzeichnis
 - `$IFS` **Separatorzeichen**
 - `$LOGNAME` aktueller login-Name
 - `$RANDOM` $0 < \text{Zufallszahl} < 32767$
 - `$OSTYPE` Betriebssystem
 - `$?` **Rückgabewert**

Variable

- ▶ Variable anlegen/setzen:
`TEST="Textinhalt"`
- ▶ Variablenwert auslesen:
`echo $TEST`
`Textinhalt`
- ▶ **Achtung:**
 - Gross- und Kleinschreibung ist relevant!
 - Variablennamen normalerweise immer in GROSSBUCHSTABEN
 - Eine nichtexistente Variable wird als "leer" betrachtet: Kein Fehler!

nichtexistente Variable

- ▶ Behandlung nichtexistenter Variablen;
Standard: "leer"
 - `${variable=wert}`
 - existiert die Variable nicht, so wird sie angelegt und ihr der Wert `wert` zugewiesen
 - `${variable-wert}`
 - existiert die Variable nicht, so wird `wert` zurückgegeben
 - `${variable+wert}`
 - existiert die Variable nicht, so "leer", sonst `wert`

Variable deklarieren

- ▶ Variablen können deklariert, typisiert und schreibgeschützt werden:
 - **declare** **NAME**
 - nicht zwangsläufig notwendig, aber auch:
 - **declare** **NAME="Vorgabewert"**
 - festlegen eines Startwertes
 - **declare -r** **NAME**
 - readonly-Variable (für immer!)
 - **declare -i** **NAME**
 - typisieren als **integer**; aufheben mit **+i**

Variablenwerte einlesen

- ▶ Variablen können Werte zugewiesen werden, die aus der Standardeingabe gelesen werden:
 - **read** **ANSWER**
 - **echo** "Ihre Eingabe war: **\$ANSWER**"
- ▶ Einlesen mehrerer Werte an verschiedene Variablen in einer Anweisung möglich; Auftrennung der Werte durch Separatorzeichen **IFS**

Felder

- ▶ Anlegen eines Feldes:
`MY=("ein" "bash" "feld" [5]="x")`
- ▶ Zugreifen auf Feldelemente:
 - ausgeben: `echo ${MY[2]}`
 - ändern/anlegen: `MY[3]="y"`
- ▶ alle Elemente ausgeben:
 - eine Zeichenkette: `echo $MY[*]`
 - in Teilen: `echo $MY[@]`
- ▶ Felder wachsen dynamisch

Arithmetik

- ▶ Eine Zeichenkette wird als arithmetischer Ausdruck ausgewertet, wenn er in doppelten runden Klammern steht:

`((arithmetischer Ausdruck))`

- ▶ `C=2*3` liefert den Wert `"2*3"`
(sofern C nicht vom Typ `integer` ist)
- ▶ `((C=2*3))` wie gewünscht `"6"`
(auch wenn C nicht vom Typ `integer` ist)

Operatoren

=	Zuweisung
+ -	unäres Plus/Minus, Addition/Subtraktion
* / %	Multiplikation, ganzzahlige Division, Restbildung
++ --	Inkrement und Dekrement als Prä- und Postfix
< <= <> >= >	Vergleichsoperatoren
== !=	Gleich und Ungleich
&&	Logisches UND/ODER
*= += %= -= /=	Zuweisungen

Bedingungen

► Kommando `test` mit vielen Optionen:

`-test 10 -eq 9`
liefert als Rückgabewert **FALSE**
– Abkürzung: `[10 -eq 9]`

► Zahlen:

<code>-eq</code>	Gleichheit	(equal)
<code>-lt</code>	kleiner	(less then)
<code>-le</code>	kleinergleich	(less equal)
<code>-gt</code>	größer	(greater then)
<code>-ge</code>	größergleich	(greater equal)

Bedingungen: test

▶ Zeichenketten:

- = Gleichheit
- != Ungleichheit
- -z Länge ist Null (zero)
- -n Länge ist größer Null

▶ Verknüpfungen:

- -a AND Ausdrücke dürfen mit
 (und) verschachtelt
 geklammert werden
- -o OR
- ! Negation

Bedingungen: test

▶ Dateien:

- -e Datei existiert (exists)
- -f ist normal (regulär)
- -d ist ein Verzeichnis
- -L ist ein symbolischer Link
- -r kann vom Benutzer gelesen werden
- -w ... geschrieben werden
- -x ... ausgeführt werden
- -s ist größer als null Zeichen