Grundlagen und Praxis der Bash- und C-Programmierung in Unix/Linux

Ein Arbeits- und Übungsbuch

Roumiana Antonova
Vassilena Slaveva
Tihomira Slavova

Pearson

Zugangscode

Falls Sie beim Kauf Ihres eBooks keinen Zugangscode erhalten haben, kontaktieren Sie uns bitte über die folgende Seite und halten Sie Ihre Rechnung/Bestellbestätigung bereit:
https://www.pearson.de/ebook-zugangscode

7.1. Skripte

Ein Skript, auch Shell- oder Bash-Skript genannt, ist eine Textdatei, die eine Folge von Befehlen, bedingten Konstrukten, Funktionen usw. enthält [5]. Es wird oft zur Beschreibung einer Aufgabe verwendet, die dann an verschiedenen Orten und/oder zu verschiedenen Zeiten ausgeführt werden kann. Das Ausführen eines Skripts führt alle darin enthaltenen Anweisungen aus. Wenn die Skriptdatei eine ausführbare Datei ist, kann sie über ihren absoluten oder relativen Pfad ausgeführt werden. Die Ausführung eines Skripts kann auch über die Befehle `bash` oder `sh` erfolgen.

Zum Beispiel kann ein Skript mit dem Namen *mein_erstes_skript.sh* im Verzeichnis */home/meinnutzer* auf eine der folgenden Arten ausgeführt werden [5]:

1. Durch Angabe der Shell
   a. mit dem Befehl `bash`:

```
bash mein_erstes_skript.sh
```

   b. mit dem Befehl `sh`:

```
sh mein_erstes_skript.sh
```

2. Wenn die Datei ausführbar ist
   a. mit dem absoluten Pfad:

```
/home/meinnutzer/mein_erstes_skript.sh
```

   b. mit relativem Pfad:

```
cd /home/meinnutzer
./mein_erstes_skript.sh
```

Ausführbare Dateien sind Dateien mit festgelegten Ausführungsberechtigungen. Weitere Informationen zu Berechtigungen finden Sie in Kapitel 11.

Es ist gute Praxis, die erste Zeile eines Skripts mit `#!` (genannt Shebang) zu beginnen, gefolgt vom Pfad der Shell, die das Skript ausführen soll [4]. Die Angabe einer Shell mithilfe eines Shebang bedeutet, dass das Skript unabhängig von der Shell des Benutzers ausgeführt werden kann [4].

7.2. Metazeichen in Dateinamen

Eine häufige Aufgabe bei der Arbeit mit dem Terminal besteht darin, nach einer Datei (oder mehreren Dateien) zu suchen, ohne deren genauen Namen oder Speicherort zu kennen, wie beispielsweise die Suche nach allen Dateien im aktuellen Verzeichnis, die im Namen die Zeichenfolge »init« am Ende, am Anfang oder in der Mitte enthalten. Diese Aufgaben können mit den Shell-Metazeichen erledigt werden – Sonderzeichen, die interpretiert werden, statt direkt an den verwendeten Befehl übergeben zu werden.

▶ Tabelle 7.1 enthält einige der Shell-Metazeichen, die häufig bei der Suche nach Mustern in Dateinamen verwendet werden [2].

Zeichen	Definition
?	Entspricht genau einem beliebigen Zeichen.
*	Entspricht einer beliebigen Anzahl von beliebigen Zeichen.
[]	Entspricht einem der Zeichen in den eckigen Klammern.

Tabelle 7.1. Shell-Metazeichen

7.3. Anführungs- und Escapezeichen

Viele Zeichen und Wörter haben eine besondere Bedeutung oder sind reserviert. Diese Beschränkung kann durch die Verwendung von Anführungs- und Escapezeichen, wie in ▶ Tabelle 7.2 beschrieben, aufgehoben werden [14].

Zeichen	Definition
Escapezeichen (\)	Hebt die besondere Bedeutung des nächsten einzelnen Zeichens auf.
Einfache Anführungszeichen (' …')	Zeigt alle Zeichen so an, wie sie sind. Einfache Anführungszeichen können nicht verschachtelt werden.
Doppelte Anführungszeichen (" …")	Zeigt alle Zeichen so an, wie sie sind, außer das Dollarzeichen, die Anführungszeichen und den Backslash.

Tabelle 7.2. Anführungs- und Escapezeichen

7.4. Variablen

Wie auch in anderen Programmiersprachen üblich, erlaubt Bash die Verwendung von Variablen. Eine Variable ist ein Speicher, verbunden mit einer Information, auf die über einen symbolischen Namen zugegriffen wird. Eine Variable kann für eine Zahl, ein Zeichen, eine Zeichenkette (String) oder ein Array stehen. Eine in Bash deklarierte Variable hat weder einen Datentyp noch darf ihr Name mit einer Zahl beginnen. Die Zuweisung eines Wertes an eine Variable erfolgt durch das Gleichheitszeichen (=) [13].

```
1 meine_string_var="Hallo Welt!"
```

Diese Codezeile definiert eine Variable namens *meine_string_var* und weist ihr den Wert »Hallo Welt!« zu. Hier werden die doppelten Anführungszeichen verwendet, um das Leerzeichen in »Hallo Welt!« zu berücksichtigen [13]. Ansonsten könnten sie weggelassen werden.

Hinweis: Bei der Definition einer Variablen sollten vor und nach dem Gleichheitszeichen keine Whitespace-Zeichen stehen [13].

Der Zugriff auf den Wert einer Variablen erfolgt über das Dollarzeichen ($). Dies wird als Parametererweiterung bezeichnet und hat die folgenden Syntaxformen [13]:

$**variable**

${**variable**}

Beispiel:

```
1 echo $meine_string_var
```

Diese Codezeile gibt den Wert von *meine_string_var* aus, z. B. `Hallo Welt!`.

7.5. Mit Variablen arbeiten

Alle Variablen der aktuellen Shell-Sitzung, einschließlich der Umgebungsvariablen, können mit dem Befehl `set` angezeigt werden, wenn er ohne Optionen und/oder Argumente ausgeführt wird [14]. Wird er mit Optionen ausgeführt, kann der `set`-Befehl auch zum Setzen oder Aufheben von Shell-Attributen verwendet werden.

▶ Tabelle 7.3 beschreibt einige der häufig verwendeten Umgebungsvariablen [5, 13].

Variable	Definition
HOME	Das Homeverzeichnis des aktuellen Benutzers
PATH	Eine Liste von durch Doppelpunkt getrennten Verzeichnissen, in denen das System nach Befehlen sucht
PS1	Der Hauptstring des Prompts (Eingabeaufforderung genannt). Diese Variable legt fest, wie der Prompt beim Starten einer Shell-Sitzung aussieht.
PS2	Der sekundäre String des Prompts. Er wird verwendet, wenn sich ein Befehl über mehrere Zeilen erstreckt.
PWD	Das aktuelle Arbeitsverzeichnis
USER	Der Name des aktuellen Benutzers
SHELL	Definiert die Shell, die alle ausgeführten Befehle interpretiert. Standard ist normalerweise Bash
HOSTNAME	Der Name des aktuellen Hostrechners.

Tabelle 7.3. Umgebungsvariablen

Das Hinzufügen von Argumenten in einer Reihenfolge definiert Werte für Positionsparameter [3].

```
set eins zwei drei
```

Ein Positionsparameter ist ein Parameter, der durch eine oder mehrere Ziffern dargestellt wird, mit Ausnahme der Ziffer 0. Positionsparameter werden zu Beginn einer Shell-Sitzung gesetzt [3]. Bei der Ausführung eines Shell-Skripts oder einer Funktion werden die Positionsparameter temporär durch die Argumente ersetzt, mit denen das Skript oder die Funktion ausgeführt wird.

```
./mein_zweites_skript.sh arg1 arg2
```

Auf den Wert eines Positionsparameters kann wie auf eine Standardvariable mit dem Dollarzeichen ($) zugegriffen werden. Positionsparameter werden der Reihe nach definiert, d. h. auf den Wert des ersten Arguments kann über $1 zugegriffen werden, auf den Wert des zweiten Arguments über $2 usw.

▶ Tabelle 7.4 beschreibt Variablen, die beim Arbeiten mit Positionsparametern verwendet werden [3].

Variable	Definition
$0	Speichert den ausgeführten Befehl, das Skript oder die Funktion in der aktuellen Shell-Sitzung.
$1 …$n	Speichert einen Wert eines Positionsparameters. Positionsparameter mit mehr als einer Ziffer können mit geschweiften Klammern erweitert werden (z. B. ${14}).
$#	Speichert die Anzahl der Positionsparameter.
$*	Speichert alle Positionsparameter. Wenn $* in doppelten Anführungszeichen verwendet wird, erweitert es die Positionsparameter zu einem einzelnen Wort.
$@	Speichert alle Positionsparameter. Wenn $@ in doppelten Anführungszeichen verwendet wird, erweitert es die Positionsparameter zu einer Liste separater Wörter, als Trennzeichen dienen die enthaltenen Leerzeichen.

Tabelle 7.4. Positionsparameter

Positionsparameter können durch den Befehl `shift` auch verändert werden. Der Befehl nimmt ein Argument in Form einer Zahl entgegen und verschiebt alle Positionsparameter um diese Zahl *N* nach links bzw. um 1, wenn keine Zahl angegeben ist [3]. Das bedeutet, dass er verwendet werden kann, um ab Anfang der Parameterliste *N* Positionsparameter zu verwerfen. Zum Beispiel führt der Code

```
set eins zwei drei vier fuenf sechs
shift 2
echo $1 # prints "drei"
```

dazu, dass die Strings *eins* und *zwei* verworfen werden, wodurch String *drei* den Wert von $1 und String *vier* den Wert von $2 erhält.

Variablen und Funktionen können mit dem Befehl `unset` entfernt werden [3]. Zum Beispiel:

```
unset meine_string_var
```

löscht die Variable *meine_string_var* und ihren Wert.

Eine Variable kann mit dem Befehl `export` an beliebige neue Prozesse übergeben werden [5]. Der Befehl kennzeichnet eine oder mehrere Variablen zur Vererbung. Wird er ohne Parameter ausgeführt, dann zeigt er die Liste aller aktuell exportierten Variablen an. Beim Exportieren einer Variablen kann der Befehl ihr auch einen Wert zuweisen. Zum Beispiel:

```
1 export exportierte_var="Ich bin exportiert!"
```

Zusammenfassung

- ✓ Ein Skript ist eine Textdatei, die eine Folge von Befehlen, bedingten Konstrukten, Funktionen usw. enthält.
- ✓ Skripte werden oft verwendet, um eine Aufgabe zu beschreiben, die an verschiedenen Orten und/oder zu verschiedenen Zeiten ausgeführt werden kann.
- ✓ Metazeichen sind Sonderzeichen, die interpretiert und nicht direkt an den verwendeten Befehl übergeben werden.
- ✓ Anführungszeichen werden verwendet, um die besondere oder reservierte Bedeutung bestimmter Zeichen aufzuheben.
- ✓ Eine Variable ist ein Speicher, der mit einer Information verbunden ist, die über einen symbolischen Namen angesprochen wird.
- ✓ Eine Variable kann eine Zahl, ein Zeichen, ein String oder ein Array repräsentieren.
- ✓ Mit dem Gleichheitszeichen (`=`) wird einer Variable ein Wert zugewiesen und mit dem Dollarzeichen (`$`) vor dem Variablennamen wird auf sie zugegriffen.
- ✓ Ein Positionsparameter ist ein Parameter, der durch eine oder mehrere Ziffern dargestellt wird, abgesehen von der Ziffer 0.
- ✓ Bei der Ausführung eines Shell-Skripts oder einer Funktion werden die Positionsparameter temporär durch die Argumente ersetzt, mit denen das Skript oder die Funktion ausgeführt wird.
- ✓ Eine Variable kann mit dem Befehl `export` an beliebige neue Prozesse übergeben werden.

Übungen zu Kapitel 7

1. Schreiben Sie einen Befehl, der alle Dateien im aktuellen Verzeichnis auflistet, deren Name mit »init« beginnt, mit ».txt« endet und:

a. nur ein Zeichen dazwischen aufweist (wie *init2.txt*);

b. eine beliebige Anzahl von Zeichen dazwischen aufweist (wie *init24ldf.txt*);

c. nur ein Zeichen dazwischen aufweist, das »a«, »b« oder »c« lautet (wie *inita.txt*, *initb.txt*, *initc.txt*).

2. Schreiben Sie einen Befehl, der alle Dateien im aktuellen Verzeichnis auflistet, die

a. Bash-Dateien sind und deren Namen mit »start« beginnen;

b. die Endung ».pp« haben;

c. Dateinamen haben, die mit »pstree« beginnen, mit »x11« enden und nur ein Zeichen dazwischen aufweisen;

d. Dateinamen mit einer Länge von 9 Zeichen haben und mit »gtf.jpeg« enden;

e. Dateinamen haben, die mit einem zufälligen Zeichen beginnen, gefolgt von dem String »art« und enden mit der Endung ».txt«;

f. *car.py*, *cat.py* oder *can.py* lauten.

3. Welche Kombination von Dateinamen kann durch Ausführen der folgenden Befehle aufgelistet werden?

a.
```
ls ar[bet]
```

b.
```
ls ba[rt]e[dr]
```

4. Das aktuelle Verzeichnis enthält die folgenden Dateien:

```
banger
barged
barding
barde
bearding
```

Wie lautet das Ergebnis nach Ausführen der folgenden Befehle?

a.
```
ls *arding
```

b.
```
ls ba[rn]ge[dr]
```

c.
```
ls bard*
```

d.
```
ls ban?er
```

5. Wie lautet das Ergebnis nach Ausführen der folgenden Befehle?

a.
```
echo "Mein Hostname ist $HOSTNAME."
```

b.
```
echo 'Mein Hostname ist $HOSTNAME.'
```

c.
```
echo "Mein Hostname ist \$HOSTNAME."
```

6. Erstellen Sie eine Variable namens *erfinder_von_linux* und setzen Sie ihren Wert auf die Zeichenkette »Linus Torvalds«. Wie lautet das Ergebnis nach Ausführen der folgenden Befehle?

a.
```
echo $erfinder_von_linux
```

b.
```
echo erfinder_von_linux
```

7. Fügen Sie */usr/X11R6/bin* zur Umgebungsvariablen `PATH` hinzu.

8. Ändern Sie Ihr aktuelles Verzeichnis in */opt*, indem Sie einen neuen Wert für die Variable `PWD` setzen.

9. Ändern Sie den Wert von `PS1` so, dass der Prompt den aktuellen Benutzernamen, den Hostnamen, das Datum (Stunden, Minuten und Sekunden) und das Verzeichnis anzeigt.

10. Zeigen Sie die aktuelle Liste der Umgebungsvariablen an. Fügen Sie eine neue Variable *STR* mit dem Wert »Hallo Welt!« hinzu und prüfen Sie, ob sie in die Liste der Umgebungsvariablen aufgenommen wurde.

11. Wie lautet das Ergebnis nach Ausführen der folgenden Befehle?

```
set var='irgendein Wert'
echo $1
```

12. Schreiben Sie ein Skript, das `a, 3, 0` und `d` als Positionsparameter setzt. Zeigen Sie die Parameter an, um zu prüfen, ob sie korrekt gesetzt sind.

13. Schreiben Sie ein Skript, das die ersten drei Positionsparameter und die Gesamtanzahl der Positionsparameter anzeigt.

14. Erstellen Sie eine Variable namens *erstes_linux_kernel_jahr*, die den Wert 1991 hat. Wie lautet das Ergebnis nach Ausführen der folgenden Befehle?

```
echo $erstes_linux_kernel_jahr
unset erstes_linux_kernel_jahr
echo $erstes_linux_kernel_jahr
```

15. Schreiben Sie einen Befehl, um den geänderten Wert der Umgebungsvariablen `PATH` aus Übung **7.** für andere Prozesse verfügbar zu machen.

16. Schreiben Sie einen Befehl, der die Variable *neue_var* exportiert und ihr gleichzeitig einen neuen Wert (Neuer Wert!) zuweist.

17. Wie oft wird `meinevar=meinname` nach der Ausführung des folgenden Skripts angezeigt?

```
#!/bin/bash
meinevar=meinname
echo meinevar=$meinevar
export meinevar
set
unset meinevar
set
```

18. Wenn sich im aktuellen Verzeichnis ein Skript *meineapp.sh* mit der folgenden Definition befindet:

```
#!/bin/bash
echo "MEINEAPP=$MEINEAPP"
MEINEAPP=2
echo "MEINEAPP=$MEINEAPP"
```

wie lautet dann Ergebnis nach Ausführen der folgenden Befehle?

```
MEINEAPP=1
bash meineapp.sh
```

Pearson

Heute schon im Lab gewesen?

Mehr Lernerfolg – immer und überall

Nie mehr Bücher schleppen oder nach verloren gegangenen Merkzetteln suchen! Mit Ihrem persönlichen Lehrbuch als kommentierbarem eText gibt Person MyLab Ihnen die Freiheit zu lernen, wann und wo immer Sie wollen. Zu Hause am PC ebenso wie unterwegs mit dem Tablet – und das ganz ohne lästiges Schleppen.

Individualisierbar

Hinterlegen Sie wichtige Notizen oder Markierungen direkt im eText. So schaffen Sie ihr persönliches Lehrbuch, mit dem Ihnen nichts mehr entgeht.

Hilfreiche Zusatzinfos direkt verlinkt

Links oder Querverweise können Sie innerhalb des eTextes neben dem Buchtext einblenden, Glossarbegriffe und Literaturhinweise können unmittelbar in einem Pop-Up-Fenster nachgeschlagen werden.

Unterwegs einfach dabei

Lesen Sie den eText in Pearson MyLab am PC, auf dem Tablet oder Smartphone. Ihr eText ist auf alle Endgeräte optimiert und bietet selbst auf kleinen Displays großen Lesekomfort!

Standardeingabe, Standardausgabe und Standardfehler; Hintergrundprozesse; Exit-Status

8

ÜBERBLICK

Copyright