

Shell-Praxis (Teil 3)



BASH
THE BOURNE-AGAIN SHELL



Vorbemerkungen

Auch dieses Praxis-Tutorial will den Umgang mit Dateien und Verzeichnissen zeigen. Insbesondere beschäftigen wir uns mit dem Kopieren, Verschieben, Umbenennen und Löschen von Dateien und Verzeichnissen.

Außerdem geht es um Shell-Skripte, die PATH-Variable und um die Initialisierungsdateien der `bash` (`~/ .bashrc` und `~/ .bash_aliases`).

Start

Zuerst aktualisieren wir unser System mit `apt`.

Die Ausgabe von `apt` wollen wir nach dem System-Update nicht mehr sehen. Wir löschen den Terminal-Inhalt mit `clear`.

```
sudo apt update && apt list --upgradable && sudo apt upgrade
[sudo] Passwort für hermann:
Holen:1 ...
OK:2 ...
...
2 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
hermann@debian:~$ clear
```

Wir bewegen uns in das Verzeichnis `~/my-tests` aus dem vorherigen Tutorial und lassen uns den Inhalt des Verzeichnisses anzeigen. Aktuell finden wir dort nur die Datei `users.txt`.

```
hermann@debian:~$ cd ~/my-tests
hermann@debian:~/my-tests$ pwd
/home/hermann/my-tests
hermann@debian:~/my-tests$ ls -al
insgesamt 12
drwxrwxr-x    2 hermann hermann 4096 Nov 10 11:46 .
drwxr-x---+  63 hermann hermann 4096 Nov 10 10:56 ..
-rw-rw-r--    1 hermann hermann  223 Nov  9 17:09 users.txt
hermann@debian:~/my-tests$
```

Kopieren von Dateien mit `cp`

Wir kopieren einige Systemdateien aus dem Verzeichnis `/etc` in das Unter-Verzeichnis `~/my-tests/my-etc`.

```
hermann@debian:~/my-tests$ mkdir my-etc
hermann@debian:~/my-tests$ cp /etc/hostname /etc/hosts /etc/services /etc/protocols /etc/fstab my-etc
hermann@debian:~/my-tests$ ls -l my-etc
insgesamt 28
-rw-r--r-- 1 hermann hermann  613 Nov 10 11:59 fstab
-rw-r--r-- 1 hermann hermann    7 Nov 10 11:59 hostname
-rw-r--r-- 1 hermann hermann 4091 Nov 10 11:59 hosts
-rw-r--r-- 1 hermann hermann 2932 Nov 10 11:59 protocols
-rw-r--r-- 1 hermann hermann 12813 Nov 10 11:59 services
```

Einzelne Dateien löschen mit `rm`

Wir entscheiden nun, dass wir die Dateien `protocols` und `services` doch nicht brauchen und löschen sie.

```
hermann@debian:~/my-tests$ rm my-etc/protocols my-etc/services
hermann@debian:~/my-tests$ ls -l my-etc
insgesamt 12
-rw-r--r-- 1 hermann hermann 613 Nov 10 12:18 fstab
-rw-r--r-- 1 hermann hermann  7 Nov 10 12:18 hostname
-rw-r--r-- 1 hermann hermann 4091 Nov 10 12:18 hosts
```

Dateibaum anzeigen mit `tree`

Das Kommando `tree` muss ggf. installiert werden mit `sudo apt install tree`.

```
hermann@debian:~/my-tests$ tree my-etc
my-etc
├── fstab
├── hostname
└── hosts

0 directories, 3 files
```

Ohne Argumente zeigt `tree` den Dateibaum des aktuellen Verzeichnisses an.

```
hermann@debian:~/my-tests$ tree
```

```
.
├── my-etc
│   ├── fstab
│   ├── hostname
│   └── hosts
└── users.txt
```

```
1 directory, 4 files
```


Verzeichnisbaum kopieren mit `cp -r`

Wir kopieren nun das Verzeichnis `/etc/apt` mit allen Unterverzeichnissen und Dateien in das Verzeichnis `my-etc`.

```
hermann@debian:~/my-tests$ cp -r /etc/apt my-etc
hermann@debian:~/my-tests$ ls -l my-etc
insgesamt 16
drwxr-xr-x 9 hermann hermann 4096 Nov 10 12:31 apt
-rw-r--r-- 1 hermann hermann  613 Nov 10 12:18 fstab
-rw-r--r-- 1 hermann hermann    7 Nov 10 12:18 hostname
-rw-r--r-- 1 hermann hermann 4091 Nov 10 12:18 hosts
hermann@debian:~/my-tests$ tree my-etc/apt
tree my-etc/apt
my-etc/apt
├── apt.conf.d
│   ├── 00aptitude
│   ...
│   └── 01autoremove
│   ...
8 directories, 19 files
```

`my-etc/apt` **recursiv löschen mit** `rm -rv`

Wir löschen das Verzeichnis `my-etc/apt` mit dem Kommando `rm`. Die Löschung erfolgt normalerweise stillschweigend, nur Fehlermeldungen werden ggf. angezeigt.

- Die Option `-r` (recursive) sorgt dafür, dass auch Unterverzeichnisse und Dateien gelöscht werden.
- Die Option `-v` (verbose) sorgt dafür, dass die gelöschten Dateien angezeigt werden.

```
hermann@debian:~/my-tests$ rm -rv my-etc/apt
...
'my-etc/apt/sources.list' wurde entfernt
Verzeichnis 'my-etc/apt' wurde entfernt
```

In der Ausgabe erscheint eine Meldung für jede gelöschte Datei und für jedes gelöschte Verzeichnis.

`/etc/apt` nochmals kopieren mit `cp -rv`

Auch das `cp`-Kommando unterstützt die Option `-v` (verbose), sodass man den Kopiervorgang dateiweise verfolgen kann.

```
hermann@debian:~/my-tests$ cp -rv /etc/apt my-etc
'/etc/apt' -> 'my-etc/apt'
...
'/etc/apt/auth.conf.d' -> 'my-etc/apt/auth.conf.d'
'/etc/apt/keyrings' -> 'my-etc/apt/keyrings'
'/etc/apt/preferences.d' -> 'my-etc/apt/preferences.d'
'/etc/apt/sources.list.d' -> 'my-etc/apt/sources.list.d'
'/etc/apt/sources.list' -> 'my-etc/apt/sources.list'
```

Aliase dauerhaft verfügbar machen

Wir haben im vorherigen Tutorial Aliase definiert, die nur für die aktuelle Shell-Sitzung gelten.

Um Aliase dauerhaft verfügbar zu machen, müssen wir sie in die Datei `~/.bash_aliases` eintragen.

Textdateien können mit jedem Texteditor bearbeitet werden. Wir verwenden `nano`. Dieser Editor ist einfach und intuitiv zu bedienen. Er ist anfängertauglich und wird auf fast allen Linux-Systemen mitgeliefert.

```
hermann@debian:~/my-tests$ nano ~/.bash_aliases
```

Schreiben Sie die folgenden (oder Ihre eigenen) Aliase in die Datei `~/.bash_aliases` und speichern Sie die Datei mit `Strg+O` und `Enter`. Beenden Sie `nano` mit `Strg+X`.

```
alias a=alias
alias cl=clear
alias h=history
alias l='ls -l'
alias la='ls -al'
alias cx='chmod u+x'
alias path='echo $PATH'
alias ssdn='sudo shutdown now'
alias srbn='sudo reboot now'
```

Kontrollieren Sie mit `cat`, ob die Aliase korrekt in der Datei `~/.bash_aliases` eingetragen sind.

Allein durch den Eintrag in der Datei `~/.bash_aliases` sind die Aliase noch nicht aktiv. Sie müssen dafür sorgen, dass `~/.bash_aliases` neu geladen wird.

1. Sie schließen die aktuelle Terminal-Sitzung und öffnen eine neue. Beim Start der neuen Sitzung wird `~/.bash_aliases` automatisch geladen. Die Aliase sind dann verfügbar.
2. Mit dem Kommando `source ~/.bash_aliases` laden Sie die Aliase in die aktuelle Shell. Das funktioniert auch ohne Beenden und Neustart der Shell.


```
hermann@debian:~/my-tests$ source ~/.bash_aliases
```

```
hermann@debian:~/my-tests$ a
alias a='alias'
alias cl='clear'
alias h='history'
alias l='ls -l'
alias la='ls -al'
alias cx='chmod u+x'
alias path='echo $PATH'
alias ssdn='sudo shutdown now'
alias srbn='sudo reboot now'
```

```
hermann@debian:~/my-tests$ a path
alias path='echo $PATH'
```

- `a` ist jetzt ein Alias für `alias`.
- `a` allein zeigt alle Aliase an.
- `a path` zeigt die Definition des Aliases `path`.

```
hermann@debian:~/my-tests$ l
insgesamt 8
drwxrwxr-x 3 hermann hermann 4096 Nov 10 13:05 my-etc
-rw-rw-r-- 1 hermann hermann  223 Nov  9 17:09 users.txt
hermann@debian:~/my-tests$ path
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
hermann@debian:~/my-tests$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Shell-Skripte

Shell-Skripte sind Textdateien, die eine Abfolge von Shell-Kommandos enthalten. Sie können mit jedem Texteditor - wir nehmen `nano` - erstellt und bearbeitet werden.

Das Ziel von Shell-Skripten ist es, wiederkehrende Aufgaben zu vereinfachen und zu automatisieren oder lange Befehlsfolgen durch einen Skript-Aufruf zu ersetzen.

Shell-Skript `system-update`

Hier wollen wir ein Shell-Skript erstellen, das den Aufruf von `sudo apt update && apt list --upgradable && sudo apt upgrade` vereinfacht und durch einen einzigen Befehl ersetzt: `system-update`.

Also erstellen wir (im Verzeichnis `my-scripts`) eine Datei `system-update`, die genau diese Befehlsfolge enthält. Wir dürfen die lange Befehlsfolge nach dem `&&` umbrechen. Einrückungen sind semantisch nicht relevant, sie dienen jedoch der besseren Lesbarkeit des Scripts.

```
sudo apt update &&  
  apt list --upgradable &&  
  sudo apt upgrade
```

```
hermann@debian:~/my-tests$ mkdir my-scripts  
hermann@debian:~/my-tests$ cd my-scripts  
hermann@debian:~/my-tests/my-scripts$ nano system-update  
hermann@debian:~/my-tests/my-scripts$ cat system-update  
sudo apt update &&  
  apt list --upgradable &&  
  sudo apt upgrade  
hermann@debian:~/my-tests/my-scripts$ l system-update  
-rw-rw-r-- 1 hermann hermann 66 Nov 10 18:54 system-update  
hermann@debian:~/my-tests/my-scripts$ bash system-update # invoke script  
[sudo] Passwort für hermann:  
...
```

Shell-Kommentare

Shell-Kommentare beginnen mit einem `#` und erstrecken sich bis zum Zeilenende. Sie dienen der Dokumentation des Skripts und sind funktional irrelevant. (Mit `nano` fügen wir Kommentare in das Skript `system-update` ein.)

```
# system-update
#
# updates all packages of the system

sudo apt update &&
  apt list --upgradable &&
  sudo apt upgrade
```

Ausführung von Skripten

Shell-Skripte können auf verschiedene Weisen ausgeführt werden:

1. mit dem Befehl `bash` und dem Skriptnamen als Argument:

```
bash system-update
```

2. mit dem Skriptnamen, dem `./` vorangestellt sein muss (Das Skript muss dazu ausführbar sein.): `./system-update`

Allgemeiner: Im Skript-Aufruf muss ein `/` enthalten sein, wenn das Skript-Verzeichnis nicht im `PATH` enthalten ist. Das aktuelle Verzeichnis ist nicht im `PATH` enthalten.

Skript ausführbar machen mit `chmod`

Der Dateimodus besteht aus 10 Zeichen. (Das erste Zeichen beschreibt den Dateityp. Es ist ein `-` für eine reguläre Datei und ein `d` für ein Verzeichnis.) Die Zeichen 2-10 beschreiben die Zugriffsrechte auf die Datei in drei Tripeln. Das erste Tripel beschreibt die Zugriffsrechte des Eigentümers, das zweite Tripel die Zugriffsrechte der Gruppe und das letzte Tripel die Zugriffsrechte aller anderen Benutzer.

Mit dem Kommando `chmod` können die Zugriffsrechte einer Datei geändert werden. Die Option `u+x` setzt das Ausführungsrecht für den Eigentümer.

```
hermann@debian:~/my-tests/my-scripts$ ls -l system-update
-rw-rw-r-- 1 hermann hermann 66 Nov 10 18:54 system-update
hermann@debian:~/my-tests/my-scripts$ chmod u+x system-update
hermann@debian:~/my-tests/my-scripts$ ls -l system-update
-rwxrw-r-- 1 hermann hermann 66 Nov 10 18:54 system-update
hermann@debian:~/my-tests/my-scripts$ ./system-update
[sudo] Passwort für hermann:
...
```

Shell-Skript `system-update` verschieben nach `~/bin`

Das Verzeichnis `~/bin` ist ein Standardverzeichnis für Benutzer-Skripte. Das Verzeichnis ist in die PATH-Variable aufzunehmen, sodass Skripte in `~/bin` ohne Pfadangabe ausgeführt werden können.

Zum Verschieben von `system-update` nach `~/bin` verwenden wir das Kommando `mv`.

```
hermann@debian:~/my-tests/my-scripts$ mkdir ~/bin
hermann@debian:~/my-tests/my-scripts$ mv system-update ~/bin
hermann@debian:~/my-tests/my-scripts$ ls -l ~/bin/system-update
-rwxrw-r-- 1 hermann hermann 66 Nov 10 18:54 /home/hermann/bin/system-update
hermann@debian:~/my-tests/my-scripts$ ~/bin/system-update # this works
[sudo] Passwort für hermann:
...
hermann@debian:~/my-tests/my-scripts$ system-update # this doesn't work
Der Befehl 'system-update' wurde nicht gefunden.
...
hermann@debian:~/my-tests/my-scripts$ echo $PATH # ~/bin is not in PATH
/home/hermann/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Die PATH-Variable

Die PATH-Variable ist eine Umgebungsvariable, die eine Liste von Verzeichnissen enthält, in denen die Shell nach ausführbaren Dateien sucht (Ausführbare Dateien können Programme oder Skripte sein.). Die Verzeichnisse sind durch Doppelpunkte getrennt. Die Reihenfolge der Verzeichnisse ist entscheidend. Die Shell sucht in den Verzeichnissen von links nach rechts nach ausführbaren Dateien.

Bei Debian-Linux ist `~/bin` normalerweise nicht in der PATH-Variablen enthalten. Wir fügen es jedoch hinzu.

Die PATH-Variable erweitern (aktuelle Shell)

```
hermann@debian:~/my-tests/my-scripts$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
hermann@debian:~/my-tests/my-scripts$ PATH=$PATH:~/bin
hermann@debian:~/my-tests/my-scripts$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin
hermann@debian:~/my-tests/my-scripts$ path
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin
hermann@debian:~/my-tests/my-scripts$ system-update # works now
[sudo] Passwort für hermann:
...
```

Wir haben die PATH-Variable in der aktuellen Shell erweitert. Die Änderung ist nur in der aktuellen Shell-Sitzung wirksam.

Die PATH-Variable erweitern (dauerhaft)

- Die Datei `~/.bashrc` wird beim Start einer neuen Shell-Sitzung geladen. Sie ist also ein geeigneter Ort, um die PATH-Variable dauerhaft zu erweitern.
- Wir fügen die Zeile `PATH=$PATH:~/bin` in die Datei `~/.bashrc` (am besten am Ende) ein.
- Dazu können einen Texteditor (wie `nano`) verwenden.
- Wir erreichen dies auch mit einer Ausgabeumlenkung mit `>>`. Der gesamte Ausdruck `PATH=$PATH:~/bin` ist dabei in einfache Anführungszeichen zu setzen.

```
hermann@debian:~/my-tests/my-scripts$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
hermann@debian:~/my-tests/my-scripts$ echo 'PATH=$PATH:~/bin' >> ~/.bashrc
hermann@debian:~/my-tests/my-scripts$ tail -1 ~/.bashrc
PATH=$PATH:~/bin
hermann@debian:~/my-tests/my-scripts$ source ~/.bashrc # reload .bashrc
hermann@debian:~/my-tests/my-scripts$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin
hermann@debian:~/my-tests/my-scripts$ system-update # works now
[sudo] Passwort für hermann:
...
```

Im Beispiel haben wir die PATH-Variable in der Datei `~/.bashrc` erweitert und sogleich mit dem Kommando `source ~/.bashrc` neu geladen.

Alternativ hätten wir auch die aktuelle Shell-Sitzung beenden und eine neue starten können. Beim Start jeder neuen `bash`-Sitzung wird `~/.bashrc` automatisch geladen.

Shell-Skript `system-update` verbessern

So sieht unser Shell-Skript `system-update` bisher aus:

```
# system-update
#
# update all packages of the system

sudo apt update &&
  apt list --upgradable &&
  sudo apt upgrade
```

Die Ausgabe der drei Kommandos ist nicht getrennt und deshalb recht unübersichtlich.

Wir verbessern das Skript, indem wir die Ausgabe der drei Kommandos voneinander trennen. Dazu fügen wir zwischen den `apt`-Kommandos den Befehl `echo` ein, der eine Leerzeile ausgibt.

```
# system-update
#
# update all packages of the system

sudo apt update &&
echo &&
apt list --upgradable &&
echo &&
sudo apt upgrade &&
```

```
hermann@debian:~/my-tests/my-scripts$ system-update
[sudo] Passwort für hermann:
...
Statusinformationen werden eingelesen... Fertig
Alle Pakete sind aktuell.

Auflistung... Fertig

Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Paketaktualisierung (Upgrade) wird berechnet... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
```

Shebang

Der Shebang ist eine Zeile am Anfang eines Skripts, die den Interpreter angibt, der das Skript ausführen soll. Der Shebang beginnt mit `#!` und wird gefolgt vom absoluten Pfad des Interpreters.

Die erste Zeile des Skripts `system-update` muss dann so aussehen:

```
#!/usr/bin/bash
```

Diese Zeile sagt der Shell, dass das Skript mit der `/usr/bin/bash` ausgeführt werden soll. Dies geschieht allerdings auch ohne Shebang.

Falscher Shebang zum Test

Wir tragen nun absichtlich das Kommando `nl -ba` als Shebang ein:

```
#!/usr/bin/nl -ba
```

In diesem Fall ruft die Shell `nl -ba` und füttert die Standardeingabe dieses Kommandos mit dem Inhalt der Datei `system-update`.

Dieser Aufruf hat die nummerierte Ausgabe der Script-Datei als Ergebnis.

```
hermann@debian:~/my-tests/my-scripts$ ./system-update
```

```
1  #!/usr/bin/nl -ba  
2  #  
3  # system-update  
4  #  
5  # update all packages of the system  
6  
7  sudo apt update &&  
8    echo &&  
9    apt list --upgradable &&  
10   echo &&  
11   sudo apt upgrade
```

Shebang wieder korrigieren

Wir korrigieren den Shebang und machen das Skript damit wieder zu einem `bash`-Skript:

```
#!/usr/bin/bash
```

Mit dem Shebang kann man den Interpreter auswählen, der das Skript ausführen soll, z.B. eine andere Shell oder auch den Interpreter für eine andere Programmiersprache wie `python3` oder `perl` etc. Die Script-Datei muss dann natürlich Code enthalten, der in der entsprechenden Sprache geschrieben ist.

Python-Skript `hello.py` mit Shebang

```
#!/usr/bin/python3  
  
print("Hello, World!")
```

```
hermann@debian:~/bin $ cat hello.py  
#!/usr/bin/python3  
  
print("Hello, World!")  
hermann@debian:~/bin $ chmod u+x hello.py  
hermann@debian:~/bin $ hello.py  
Hello, World!
```


Zusammenfassung

In Fortsetzung der vorherigen Tutorials haben wir uns mit dem Kopieren, Verschieben, Umbenennen und Löschen von Dateien und Verzeichnissen und ganzen Verzeichnisbäumen beschäftigt. Auch das Anlegen von Shell-Skripten und die Verwendung von Aliase und der PATH-Variablen waren Thema.

Hier nochmals die Auflistung der wichtigsten Themen und Befehle:

- `cp` zum Kopieren von Dateien und Verzeichnisbäumen mit den Optionen `-r` (recursive) und `-v` (verbose)

- `rm` zum Löschen von Dateien und Verzeichnisbäumen mit den Optionen `-r` (recursive) und `-v` (verbose)
- `tree` zum Anzeigen von Verzeichnisbäumen
- `nano` Editor zum Bearbeiten von Textdateien
- Aliase dauerhaft verfügbar machen in `~/.bash_aliases`
- Shell-Skripte erstellen und ausführbar machen mit `chmod` am Beispiel des Scripts `system-update`
- Kommentare in Shell-Skripten (und am Prompt) mit `#`
- Verschieben von Dateien und Verzeichnissen mit `mv`
- Die PATH-Variable dauerhaft erweitern in `~/.bashrc`
- Shebang am Anfang von `bash`-Skripten: `#!/usr/bin/bash`