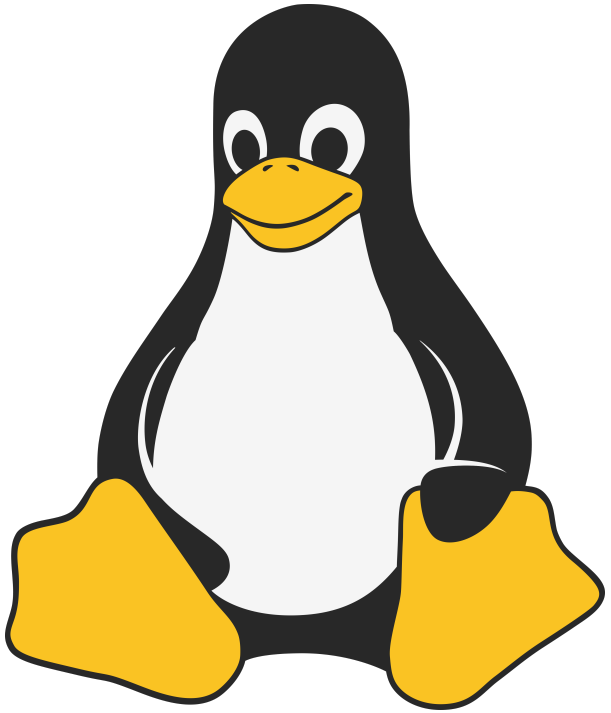
Shell-Praxis

Shell-Skripte



Inhaltsverzeichnis

Shell-Skripte: Was sind sie?

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.

Zum Inhaltsverzeichnis ...

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.

Zum Inhaltsverzeichnis ...

```
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 irrelavant. (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
```


Zum Inhaltsverzeichnis ...

Ausführung von Skripten (`chmod`)

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 eine Pfadangabe vorangestellt sein muss (Das Skript muss dazu ausführbar sein.): `./system-update` (wenn das Skript im aktuellen Verzeichnis liegt)

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

Zum Inhaltsverzeichnis ...

Skript ausführbar machen mit `chmod`

Der Dateimodus besteht aus 10 Zeichen. (Das erste Zeichen beschreibt den Dateityp. Es ist ein `-` für eine reguläre Datei, ein `d` für ein Verzeichnis oder ein `l` für einen symbolischen Link.) 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.

Zum Inhaltsverzeichnis ...

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:
...
```


Zum Inhaltsverzeichnis ...

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`.

Zum Inhaltsverzeichnis ...

```
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 doen'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
```

Dateien umbenennen oder verschieben: `mv`

Das Kommando `mv <source> <destination>` verschiebt die Datei `<source>` oder benennt sie um.

- Ist `<destination>` ein Verzeichnis, wird `<source>` in dieses Verzeichnis verschoben. Der letzte Pfadnamensbestandteil von `<source>` bleibt dabei erhalten.
- Ist `<destination>` kein Verzeichnis, wird `<source>` umbenannt in `<destination>`.
- `<source>` kann eine Datei oder ein Verzeichnis sein.

Zum Inhaltsverzeichnis ...

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.

Zum Inhaltsverzeichnis ...

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=$PATH:~/bin` in die Datei `~/.bashrc` (am besten am Ende) ein.
- Dazu können einen Texteditor (wie `nano`) verwenden.
- Wir erreichen dies auch mit einer Ausgabeumlenkung mit `>>` zum Anhängen an `~/.bashrc`. **Wichtig!** Der gesamte Ausdruck `'PATH=$PATH:~/bin'` steht dabei in einfachen Anführungszeichen. (Keine Erläuterung an dieser Stelle)

```
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 durch Leerzeilen voneinander trennen. Dazu fügen wir zwischen den `apt`-Kommandos den Befehl `echo` ein, der ohne Argumente nur 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.
```

Wenn Sie wollen, können Sie auch statt der Leerzeilen jeweils eine Zeile mit einer Trennlinie einfügen.

Zum Inhaltsverzeichnis …

Shebang

Der Shebang (auch Hashbang genannt) 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:

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

Diese Zeile sagt der Shell, dass das Skript mit der `/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.

Testen Sie auch mit dem Kommando `wc` als Shebang.

Zum Inhaltsverzeichnis ...

```
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:

```
#!/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 Skript-Datei muss dann natürlich Code enthalten, der in der entsprechenden Programmiersprache geschrieben ist.

Zum Inhaltsverzeichnis ...

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!
```

Key Takeaways

- Shell-Skripte sind Textdateien, die eine Abfolge von Shell-Kommandos enthalten.
- Ein Shebang (auch Hashbang) am Anfang des Skripts gibt den Interpreter an, der das Skript ausführen soll. Ist kein Shebang vorhanden, wird das Skript mit der Standard-Shell ausgeführt.
- Shell-Skripte sollen unabhängig vom aktuellen Verzeichnis durch die Angabe des Skript-Namens (ohne Pfadangabe) ausgeführt werden können - so wie auch `ls`, `ps`, `grep` und viele andere Kommandos.

Zum Inhaltsverzeichnis ...

- Dazu müssen zwei Voraussetzungen erfüllt sein:
  - Das Skript muss ausführbar sein.
  - Das Skript-Verzeichnis muss in der PATH-Variable enthalten sein.
- Die Erweiterung der PATH-Variablen wird am zweckmäßigsten in der Datei `~/.bashrc` definiert, sodass sie bei jedem Start einer neuen Shell-Sitzung geladen wird.
- Als Skript-Verzeichnis für Benutzer-Skripte wird üblicherweise `$HOME/bin` verwendet.

Zum Inhaltsverzeichnis ...

Aufgaben

- Erstellen Sie das Shell-Skript `system-update` im Verzeichnis `my-tests/my-scripts`.
- Rufen Sie das Skript auf und übergeben der `bash` den Skriptnamen als Argument.
- Machen Sie das Skript ausführbar und rufen Sie es auf. Funktioniert dies auch ohne Angabe des (relativen oder absoluten) Skriptpfades?
- Setzen Sie die PATH-Variable in der aktuellen Shell so, dass das Skript ohne Pfadangabe ausgeführt werden kann.

Zum Inhaltsverzeichnis ...

- Erweitern Sie die PATH-Variable in der Datei `~/.bashrc` um das Verzeichnis `~/bin`.
- Beenden Sie die aktuelle Shell-Sitzung im Terminal und starten Sie eine neue. Testen Sie, ob das Skript `sytem-update` in der neuen Shell jetzt ohne Pfadangabe ausgeführt werden kann. Ist dies der Fall, dann haben Sie die PATH-Variable in `~/.bashrc` richtig erweitert.
- Ergänzen Sie das Skript `system-update` um einen Shebang, der den Interpreter `/bin/bash` angibt.

- Ergänzen Sie das Skript mit einem sinnvollen Kommentar.
- Teilen Sie die lange Befehlsfolge in drei Zeilen auf.
- Testen Sie das Skript nach jedem Veränderungsschritt.
- Der Dozent stellt einige Skripte zum Ausprobieren zur Verfügung. Kopieren Sie diese in ein neues Verzeichnis `~/bin-trainer`. Nehmen auch dieses Verzeichnis in die PATH-Variable auf machen Sie die Skripte ausführbar.
- Sehen Sie sich die Skripte an (mit `cat` oder `nl -ba`) und testen Sie sie.

Zum Inhaltsverzeichnis ...

- Prüfen Sie mit dem Kommando `which`, in welchem Verzeichnis ein Kommando liegt. In welchem Verzeichnis liegen die folgenden Kommandos:
  - `ls`
  - `grep`
  - `cowsay`
  - `fortune`
  - `system-update`
  - etc.