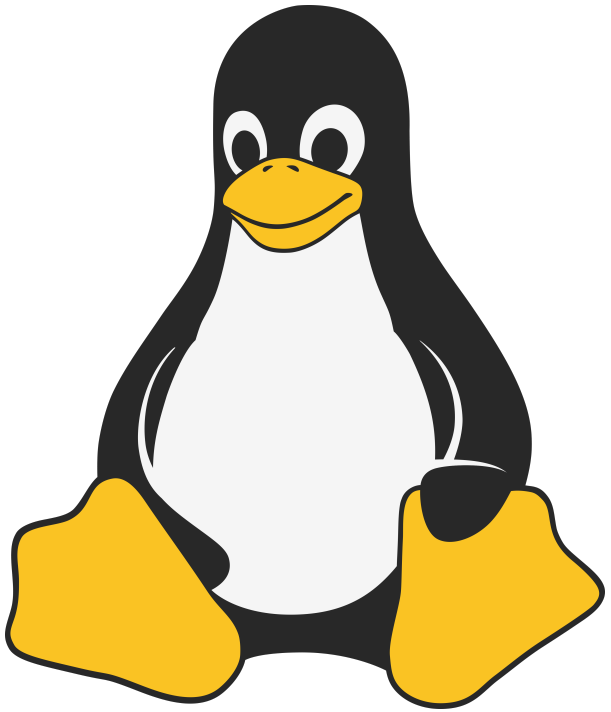
Shell-Skripte



Inhaltsverzeichnis

- Die PATH-Variable
- Shell-Skript `system-update` verbessern
- Shebang
- Key Takeaways
- Übungen

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`

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

Also legen wir das Verzeichnis `~/bin` und erstellen dort die Textdatei `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 Skripts.

Zum Inhaltsverzeichnis ...

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

```
hermann@debian:~$ mkdir bin
hermann@debian:~$ nano bin/system-update
hermann@debian:~$ nl bin/system-update
     1  sudo apt update &&
     2    apt list --upgradable &&
     3    sudo apt upgrade
hermann@debian:~$ ls -l bin/system-update
-rw-rw-r-- 1 hermann hermann 66 Nov 10 18:54 system-update
hermann@debian:~$ bash bin/system-update # invoke script
...
```

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 (absoluten oder relativen) Pfad des Skripts als Argument: `bash bin/system-update`
2. mit dem Skriptnamen, dem eine Pfadangabe vorangestellt sein muss (Das Skript muss dazu ausführbar sein.): `bin/system-update`

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

Zum Inhaltsverzeichnis ...

Skript ausführbar machen mit `chmod`

(Zu Dateirechten und `chmod` siehe auch Kapitel 04)

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

```
hermann@debian:~$ ls -l bin/system-update
-rw-rw-r-- 1 hermann hermann 66 Nov 10 18:54 system-update
hermann@debian:~$ chmod u+x bin/system-update
hermann@debian:~$ ls -l system-update
-rwxrw-r-- 1 hermann hermann 66 Nov 10 18:54 bin/system-update
hermann@debian:~$ bin/system-update
...
```


Zum Inhaltsverzeichnis ...

Der Skript-Aufruf funktioniert bisher nur, wenn das Skript mit einem absoluten oder relativen Pfad aufgerufen wird. Wir wollen das Skript jedoch auch ohne Pfadangabe ausführen können.

```
hermann@debian:~$ ~/bin/system-update # this works
[sudo] Passwort für hermann:
...
hermann@debian:~/bin$ system-update # this doesn't work
Der Befehl 'system-update' wurde nicht gefunden.
...
```

```
hermann@debian:~/bin$ echo $PATH # script dir '~/bin' IS NOT in the PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Die PATH-Variable

Damit das Skript `system-update` ohne Pfadangabe von der Shell gefunden und ausgeführt werden kann, muss das Skript-Verzeichnis `~/bin` in der PATH-Variablen enthalten sein.

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

Zum Inhaltsverzeichnis ...

Die PATH-Variable erweitern (aktuelle Shell)

```
hermann@debian:~$ echo $PATH # script dir '~/bin' IS NOT in the PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
hermann@debian:~$ PATH=$PATH:~/bin
hermann@debian:~$ echo $PATH # script dir '~/bin' IS NOT in the PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin
hermann@debian:~$ system-update # works now
...
```

Wir haben die PATH-Variable in der aktuellen Shell erweitert. Die Änderung ist nur in der aktuellen Shell-Sitzung wirksam. (Diese Änderung ist nicht dauerhaft.)

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 hängen die Zeile `PATH=$PATH:~/bin` ans Ende der Datei `~/.bashrc` an mit einer Ausgabeumlenkung zum Anhängen (`>>`).
- **Wichtig!** Der gesamte Ausdruck `'PATH=$PATH:~/bin'` muss in einfachen Anführungszeichen stehen. Warum?

Zum Inhaltsverzeichnis ...

```
hermann@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
hermann@debian:~$ echo 'PATH=$PATH:~/bin' >> ~/.bashrc
hermann@debian:~$ tail -1 ~/.bashrc
PATH=$PATH:~/bin
hermann@debian:~$ exit
Abgemeldet
```

Nach der Wiederanmeldung in der Shell-Sitzung ist die PATH-Variable erweitert.

```
hermann@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin
hermann@debian:~$ system-update # works now
...
```

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 in drei Ausgabeblöcke getrennt und deshalb recht unübersichtlich.

Zum Inhaltsverzeichnis ...

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.

Shebang

Der Shebang (auch Hashbang) ist ein Kommentar in der 1. Zeile eines Skripts, die den Interpreter angibt, der das Skript ausführen soll. Der Shebang beginnt mit `#!` und wird gefolgt vom absoluten Pfad des Interpreters.

Die 1. Zeile des Skripts `system-update` muss dann so aussehen:

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

Diese Zeile gehört nicht zum Skript-Inhalt. Sie teilt der Shell mit, dass das Skript mit der `/bin/bash` ausgeführt werden soll.

Die `bash` wird auch ohne Shebang zur Skriptausführung verwendet, denn die `bash` ist die Standard-Shell für den angemeldeten Benutzer.

- Die für den Benutzer eingestellte Standard-Shell ist in der Umgebungsvariablen `SHELL` gespeichert.

```
hermann@debian:~$ echo $SHELL
/bin/bash
```

Im Shebang eines Skripts kann auch ein anderer Interpreter eingetragen werden.

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.

Python-Skript `hello.py` mit Shebang

```
#!/usr/bin/python3

print("Hello, World!")
```

```
hermann@debian:~/bin $ nl -ba hello.py
     1  #!/usr/bin/python3
     2
     3  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 (z.B. `#!/bin/bash`) gibt den Interpreter an, der das Skript ausführen soll. Ist kein Shebang vorhanden, wird das Skript mit der Standard-Shell `$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 (x-Bit gesetzt)
  - Das Skript-Verzeichnis muss in der PATH-Variablen enthalten sein.
- Die Erweiterung der PATH-Variablen wird am zweckmäßigsten in der Datei `~/.bashrc` definiert, sodass sie bei jedem Start einer neuen Shell geladen wird.
- Als Skript-Verzeichnis für Benutzer-Skripte wird üblicherweise `$HOME/bin` verwendet.

Zum Inhaltsverzeichnis ...

Übungen

- Erstellen Sie das Shell-Skript `system-update` im Verzeichnis `~/bin` (wie in dieser Lektion gezeigt).
- 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 mit `nl -ba`) und testen Sie sie.

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