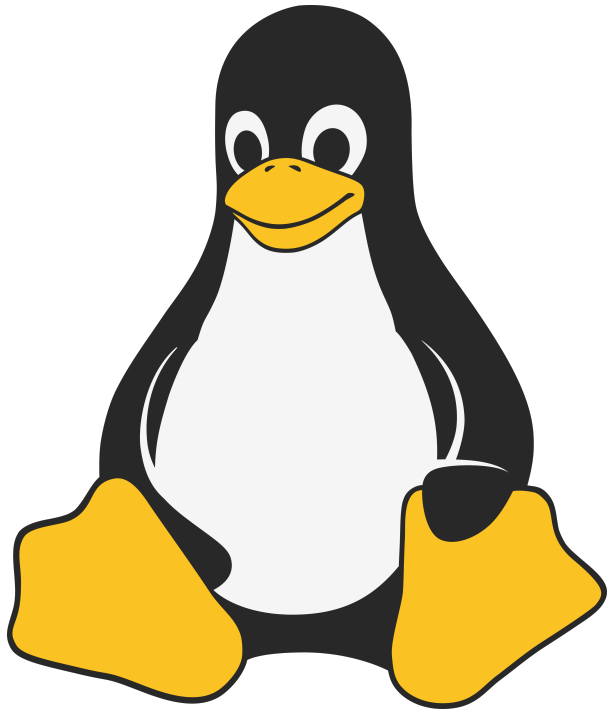
Initialisierung der Bash-Shell



Inhaltsverzeichnis

- Aufräumarbeiten beim Logout
- Bash-Initialisierung in `~/.bashrc`
  - Variablen-Definitionen `~/.profile` oder `~/.bashrc`?
  - Alias-Definitionen in `~/.bashrc` oder `~/.bash_aliases`?
- System-weite Initialisierung in `/etc/bash.bashrc`

Überblick

- Es gibt eine ganze Reihe von Dateien, die der Initialisierung der Bash-Shell dienen.
- Die Regeln der `bash`-Initialisierung sind etwas unübersichtlich.
- Wir versuchen, sie hier in den wichtigsten Punkten zu erläutern. Wir werden dabei einige Vereinfachungen aus Gründen der Übersichtlichkeit und Verständlichkeit vornehmen.

Zum Inhaltsverzeichnis ...

- Wir blicken besonders auf die benutzer-spezifischen Initialisierungsdateien, die alle im Heimatverzeichnis des Benutzers liegen und mit einem `.` beginnen.
- Die systemweit für alle Benutzer gültigen Initialisierungsdateien in `/etc`, werden nur kurz behandelt.

Zum Inhaltsverzeichnis ...

- Dabei gilt es drei Initialisierungsphasen zu unterscheiden:
  - Initialisierung beim Login (Terminal-Login, SSH-Login; gilt nicht beim Login in der grafischen Oberfläche)
  - Aufräumarbeiten beim Logout
  - Initialisierung beim Start jeder interaktiven Shell (gilt nicht für Skript-Shells)

Die Initialisierung erfolgt durch das Laden von Konfigurationsdateien. Dies sind Shell-Skripte, die beim Start der Shell geladen werden, um die Shell-Umgebung zu konfigurieren, z. B.

- Definition von Umgebungsvariablen,
- Definition von Aliasen,
- Definition von Funktionen,
- Setzen von Shell-Optionen.

Man kann auch so etwas einfaches tun wie den Bildschirm löschen oder einen Fortune-Spruch anzeigen.

Zum Inhaltsverzeichnis ...

Initialisierung beim Login

Drei benutzer-spezifische Dateien können beim Login geladen werden:

- `~/.bash_profile`: **1. Priorität**
- `~/.bash_login`: **2. Priorität**, wird geladen, falls `~/.bash_profile` nicht existiert.
- `~/.profile`: **3. Priorität**, wird geladen, falls die beiden anderen nicht existieren.

`~/.profile` - 3. Priorität

- `~/.profile` ist die Standard-Initialisierungsdatei für die alte Bourne-Shell `/bin/sh`.
- Aus Grüden der Bourne-Shell-Kompatibilität wird sie auch von der Bash berücksichtigt.
- Sie wird von der Bash geladen, wenn keine der anderen Dateien (`~/.bash_profile`, `~/.bash_login`) existiert.

Zum Inhaltsverzeichnis ...

`~/.bash_login` - 2. Priorität

- `~/.bash_login` wird von der Bash geladen, wenn `~/.bash_profile` nicht existiert.
- Sie wird nur wegen bestimmter Namenskonventionen unterstützt (Analogie zu `~/.bash_logout`, Analogie zur C-Shell: Die `csh` wurde in `.login` initialisiert).
- In der Praxis wird `~/.bash_login` kaum verwendet.
- Empfehlung: Verwenden Sie bevorzugt `~/.bash_profile` oder `~/.profile`.

`~/.bash_profile` - 1. Priorität

- `~/.bash_profile` wird von der Bash geladen, wenn sie existiert.
- Sind `~/.bash_login` und/oder `~/.profile` ebenfalls vorhanden, werden diese ignoriert.
- Sollen `~/.bash_profile` und `~/.profile` beide geladen werden, so kann `~/.profile` in `~/.bash_profile` explizit mit dem Kommando `source ~/.profile` oder `. ~/.profile` geladen werden.

Zum Inhaltsverzeichnis ...

`~/.bash_profile` oder `~/.profile` verwenden?

- Verwendet ein Benutzer gelegentlich auch die Bourne-Shell als Login-Shell, so muss er `~/.profile` verwenden.
- Verwendet ein Benutzer ausschließlich die `bash` als Login-Shell, so ist es technisch egal, ob er `~/.bash_profile` oder `~/.profile` verwendet.
- Viele Benutzer verwenden `~/.profile` "aus guter, alter Tradition".
- Viele Distributionen (auch Debian) legen `~/.profile` mit einer Standardkonfiguration an, wenn ein neuer Benutzer angelegt wird. `~/.bash_profile` fehlt.

Zum Inhaltsverzeichnis ...

System-weite Initialisierung in `/etc/profile`

- Bevor die benutzer-spezifischen Initialisierungsdateien geladen werden, wird die System-Initialisierungsdatei `/etc/profile` geladen.
- Diese Datei wird für alle Benutzer geladen, die sich an einem Terminal-Login anmelden.
- Sie kann nur mit Root-Rechten bearbeitet werden.

Zum Inhaltsverzeichnis ...

Aufräumarbeiten beim Logout

- Beim Logout wird die Datei `~/.bash_logout` ausgeführt, falls sie existiert.
- Hier können z. B. temporäre Dateien gelöscht werden.
- Typischerweise wird hier nur der Terminal-Bildschirm gelöscht, damit die letzten Kommandoeingaben nach dem Logout nicht mehr sichtbar sind. (Schutz der Privatsphäre)

Zum Inhaltsverzeichnis ...

Initialisierung einer interaktiven Shell in `~/.bashrc`

- Die Datei `~/.bashrc` wird beim Start einer interaktiven `bash` geladen, auch von einer Subshell.
- Eine Skript-Shell ist nicht interaktiv. Sie lädt `~/.bashrc` nicht.
- **Ausnahme**: `~/.bashrc` wird nicht geladen von einer Login-Shell. Eine Login-Shell lädt stattdessen eine der Dateien `~/.bash_profile`, `~/.bash_login` oder `~/.profile` (s.o.).

Zum Inhaltsverzeichnis ...

- `~/.bashrc` kann auch explizit in einer der Login-Konfigurations-Dateien mit dem Kommando `source ~/.bashrc` oder `. ~/.bashrc` geladen werden.
- Dieses Verfahren ist üblich, wie z. B. in `~/.profile` bei Debian.

```
# ~/.profile
# load ~/.bashrc if it exists
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```


Zum Inhaltsverzeichnis ...

- `~/.bashrc` enthält Befehle, die bei jedem Start einer interaktiven Bash-Shell ausgeführt werden sollen. Typischerweise sind das:
  - Definition von Umgebungsvariablen,
  - Definition von Aliasen,
  - Definition von Funktionen (werden hier nicht behandelt),
  - Setzen von Shell-Optionen (werden hier nicht behandelt).

Variablen-Definitionen `.profile` oder `~/.bashrc`?

- Variablen-Definitionen können in `~/.bashrc` oder in einer der Login-Konfigurations-Dateien, z. B. `~/.profile`, erfolgen.
- In der Regel werden sie in `~/.bashrc` definiert.
- Wird eine Variable in `~/.bashrc` definiert, so ist sie auch in allen Subshells verfügbar. Denn `~/.bashrc` wird auch von Subshells geladen.

Zum Inhaltsverzeichnis ...

- Wird eine Variable in einer der Login-Konfigurations-Dateien wie `~/.profile` definiert, so ist sie nur in der Login-Shell verfügbar.
- Wird die Variable jedoch exportiert, so ist sie auch in Subshells verfügbar.

Alias-Definitionen in `~/.bashrc` oder `~/.bash_aliases`?

- Aliase können in `~/.bashrc` oder `~/.bash_aliases` definiert werden. Technisch macht dies keinen Unterschied.
- Typischerweise werden eigene Aliase in `~/.bash_aliases` definiert.
- Dazu muss `~/.bash_aliases` in `~/.bashrc` geladen werden mit dem Kommando `source ~/.bash_aliases` oder `. ~/.bash_aliases`.
- Bei Debian Linux ist die Einbindung von `~/.bash_aliases` in `~/.bashrc` bereits enthalten.

Zum Inhaltsverzeichnis ...

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

Wenn die Datei `~/.bash_aliases` existiert, wird sie mit dem Kommando `.` in die aktuelle Shell geladen.

System-weite Initialisierung in `/etc/bash.bashrc`

- Vor dem Laden der Datei `~/.bashrc` wird die System-Initialisierungsdatei `/etc/bash.bashrc` geladen.
- Diese Datei wird für alle Benutzer geladen, die eine interaktive `bash`-Shell starten.
- Sie kann nur mit Root-Rechten bearbeitet werden.

Zum Inhaltsverzeichnis ...