Shell

Bash & Co.

- Debian-Pakete mit Status rc entfernen
- Scripte per Cronjob starten
- Upgrade Debian Bookworm zu Trixie
- Bash - Besondere Variablen
- Bash - Nützliche Links

Debian-Pakete mit Status rc entfernen

Werden Pakete mit `apt remove PAKETNAME` entfernt, bleiben die Konfigurationsdateien des Pakets im Dateisystem über.

Das lässt sich verhindern, indem `apt purge PAKETNAME` ausgeführt wird.

Nicht komplett gelöschte Pakete werden beim Auflisten, markiert durch ein rc am anfang der Zeile, angezeigt.

Anzeigen aller installierter Pakete:

```
dpkg -l
```

Ein rc am Anfang der Zeile bedeutet, dass die Pakete entfernt (r=**r**emoved) wurden, die Konfigurationsdateien (c=**c**onfiguration files) jedoch noch im System vorhanden sind.

Ermitteln aller Pakete mit Status rc:

```
dpkg -l | grep ^rc | awk '{print $2}'
```

Simulation der Löschung ermittelter Pakete:

```
dpkg --purge --simulate $(dpkg -l | grep ^rc | awk '{print $2}')
```

Löschen der Pakete:

```
dpkg --purge $(dpkg -l | grep ^rc | awk '{print $2}')
```

Scripte per Cronjob starten

Falls durch einen Cronjob gestartete Scripte nicht richtig ausgeführt werden oder andere Ergebnisse liefern als erwartet kann das an fehlenden Pfaden (PATH) in den Umgebungsvariablen des Cron liegen. Diese sind beim Ausführen von der Shell andere als bei der Ausführung durch Cron. Ein Script kann somit in der Shell ordnungsgemäß arbeiten aber Probleme beim Ausführen durch Cron haben.

Vergleich der Pfade von Shell und Cron

Pfade in der Shell

In der Shell bekommet man alle Umgebungsvariablen mit dem Befehl printenv:

```
printenv | grep PATH
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Pfade von Cron

Die chrontab editieren:

```
crontab -e
```

Ergänzen der bereits bestehenden Einträge um eine Zeile mit dem Inhalt:

```
*/1 * * * * printenv | grep PATH > /root/printenv.cron
```

Nach Ablauf einer vollen Minute kann der Eintrag aus der crontab wieder entfernen werden, da er sonst jede volle Minute erneut ausgeführt wird.

Den Inhalt der Datei printenv.cron ausgeben:

```
cat /root/printenv.cron
```

```
PATH=/usr/bin:/bin
```

Die Pfade des Cron sind nur auf /usr/bin und /bin gesetzt.

Ergänzen der Pfade für shell-Scripte in der crontab

Nun die Pfade in der crontab ergänzen. Dazu die crontab editieren und fdie Pfade vor den Cronjobs hinzufügen:

```
crontab -e
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Ergänzen der Einträge für Python-Scripte in der crontab

Für die Ausführung von Python-Scripten folgende Einträge in der crontab hinzufügen:

```
crontab -e
```

```
PYTHONPATH=$PYTHONPATH:/usr/local/lib/python3.X/dist-packages/
```

Upgrade Debian Bookworm zu Trixie

Vorbereitung der alten Version

Update des Paket Index

```
apt update
```

Aktualisieren aller Pakete

```
apt dist-upgrade
```

Ändern der Debian Paket Repositories

```
sed -i 's/bookworm/trixie/g' /etc/apt/sources.list
```

```
sed -i 's/bookworm/trixie/g' /etc/apt/sources.list.d/*
```

Upgrade auf die neue Version

Update des Paket Index auf die neue Version

```
apt update
```

Upgrade des Systems auf die neue Version

```
apt dist-upgrade
```

Bash - Besondere Variablen

Liste der besonderen Variablen

- `$0` – Der Name des ausgeführten Scripts
- `$1-$9` – Die Kommandozeilen-Argumente 1-9
- `$#` – Die Anzahl der Kommandozeilen-Argumente
- `$*` – Alle Kommandozeilen-Argumente in einem String
- `$@` – Alle Kommandozeilen-Argumente als Array
- `$?` – Der Exit-Status des letzten Kommandos
- `$$` – Die Prozess ID der aktuellen Shell
- `$!` – Die Prozess ID des letzten Hintergrundkommandos
- `$-` – Zeigt die aktuellen Shell Optionen und Flags

Beispiele

```
#!/bin/bash
echo -e "\nDer Name des Scripts lautet "$0"\n" # $0 - Name des ausgeführten Scripts
echo -e "Es wurde(n) "$#" Kommandozeilenargument(e) mit angegeben\n" # $# - Anzahl der
Kommandozeilen-Argumente
if [[ $# -ne 0 ]]
then
    x=1
    for i in $@; # $@ - Alle Kommandozeilen-Argumente als Array
    do
        echo -e "Das "$x". Kommandozeilenargument war "$i"\n"
        ((x++))
    done
    echo -e "Alle Kommandozeilenargumente waren: "$*"\n" # $* - Alle Kommandozeilen-Argumente
in einem String
fi
```

```
#!/bin/bash
ping 192.168.0.1 -c1 -W1 > /dev/null
result=$? # $? - Exitstatus des letzten Kommandos
if [[ $result -ne 0 ]]
then
    echo "Ping failed"
```

```
else
    echo "Ping OK"
fi
```

Bash - Nützliche Links

https://devhints.io/bash

https://www.w3schools.com/bash/index.php

https://linuxize.com/series/bash-scripting-fundamentals/