HOCHSCHULE ESSLINGEN

Laborversuch 1: Shellskripte

Lernziele:

In diesem Laborversuch sollen Sie lernen, Routineaufgaben auf einem Linuxrechner mit einem Bash-Skript zu automatisieren. Sie müssen hierzu die ersten Aufgaben der Vorlesung durchmachen, d.h. die Bash Cheat Sheets durcharbeiten, sowie das Skript ***Skript Linux-Crashkurs***, lesen.

Erfolgskontrolle:

Um den Versuch testiert zu bekommen, müssen Sie **vorher** die Vorbereitungsfragen in Moodle beantwortet und mindestens 50% der Punkte erreichen. Am Ende dieses Labors müssen Sie jeder ihr eigenes Skript in Moodle hochladen und erfolgreich schlüssig erklären können.

Einführung:

Ein Server läuft typischerweise 24/7, d.h. rund um die Uhr und sogenannte down-times müssen limitiert werden (Stichwort „5 niners“ mit 99.999% uptime). Das Kommando `uptime` liefert genau diese Information seit dem **letzten** Reboot. Auch ist wichtig, wer sich häufig von wo einloggt, d.h. welcher Nutzer wie häufig auf einem System zu Gange ist und **von wo**. Wie auch immer, all diese Informationen finden sich im Kommando `last`:

Ausgabe von last auf dem BS-Server:

```
rakeller pts/6        134.108.34.30    Wed Feb 22 04:17   still logged in
pkoester pts/10       134.108.63.67    Wed Feb 22 04:15   still logged in
rakeller pts/9        134.108.34.30    Tue Feb 21 15:48 - 04:17  (12:28)
soboettc pts/7        134.108.63.43    Tue Feb 21 08:06 - 09:45  (01:38)
pkoester pts/6        134.108.63.29    Tue Feb 21 08:03 - 12:23  (04:19)
reboot   system boot  5.14.0-162.12.1. Tue Jan 31 06:27   still running

wtmp begins Thu Sep 22 10:16:52 2022
```

Die Spalten führen (mit fester Anzahl Zeichen) auf:

1. Welcher Nutzer, bzw. ob ein Reboot passiert ist,
2. Auf welchem lokalen Terminal, bzw. wieder daß ein Reboot passiert ist,
3. Von welcher IP-Adresse, bei Reboot die gebootete Kernel-Version,
4. Zeitpunkt des Einloggens, bzw. des Reboot
5. Zeitpunkt des Ausloggens, mit einer Dauerangabe in Klammern, bzw. die jeweilige Uptime. Für Nutzer bekommt man noch raus, ob diese noch eingeloggt sind, für gebootete Kernel, ob diese noch laufen.

Man kann daraus also viele administrative Aufgaben erfassen: wie häufig ist ein Benutzer eingeloggt. Durchschnittlich wie lange ist dieser eingeloggt. Von welchen Maschinen loggt er sich ein. Wie lange lief ein bestimmter Kernel, wie lange am Stück war ein Server up-and-running.

Übrigens, die zwei letzten Zeilen stammen aus darunterliegenden „Datenbank“ `/var/lib/wtmp` und haben mit unserer Analyse nichts zu tun (müssen also in unserer Ausgabe eliminiert werden).

Ihre Aufgabe: schreiben Sie das bash-Skript "`last_analysis.sh`"

Ihr Script heißt `last_analysis.sh`. Es analysiert verschiedene Dinge:

- `-r` oder `--runtime` listet die Uptime von Reboots auf (s. beispielhafte Ausgabe unten),
- `-u` oder `--user` sortiert nach Anzahl Logins von Nutzern und dem Nutzernamen,

Das Skript soll nur nach einem Parameter ausgeben, nicht nach mehreren, der Default ist die Option `runtime`. Das Skript soll einen weiteren Parameter `-h` und `--help` zur Verfügung stellen, um obige Parameter zu erklären – und sich dann beenden.

Ausgabe der Runtime

Beispielhaft Ausgabe des Scriptes für `--runtime`:

```
5.14.0-162.12.1. still running
5.14.0-162.6.1.e 56 days
5.14.0-70.30.1.e 0 days
5.14.0-70.26.1.e 27 days
```

Hier ist die Uptime also immer in Tagen angezeigt, wobei der neueste Reboot sicherlich wegen dem Update auf den neuesten Kernel (5.12.0-162.12.1 passiert ist, s. Ausgabe von `uname -a`).

Für die Ausgabe ist eine geschickte Verbindung von `last`, `grep`, sowie `cut` notwendig. So bekommt man mit `last | grep -E '^reboot' | grep -v 'still running'` alle Zeilen, in denen **am Anfang der Zeile** (erweiterter Regulärer Ausdruck -E) der String `reboot` steht, von denen man alle Zeilen rausschneidet in welchen `still running` drin steht (`-v` für invert match). Probieren Sie diese Zeile aus.

Diese Ausgabe können Sie in eine Datei umleiten mittels des Befehls„>“. Und dann diese Datei wieder zeilenweise auslesen mittels `while read LINE ; do echo LINE:$LINE; done < tmp`

Ausgabe der User

Beispielhaft Ausgabe des Scriptes für `--user`:

```
12 pkoester
49 rakeller
855 root
```

Ähnlich zum obigen Vorgehen lohnt es sich, eine temporäre Datei mit den sortierten Usern zu füllen (`sort` kennt den Befehl unique mit `-u`). Sodann kann man in einer Schleife über diese Nutzer Zeile für Zeile deren Anzahl in Last greppen, und diese Zeilen zählen. Sort kennt wiederum die numerische Sortierung mit `-n`, sodaß die Nutzer mit den meisten Logins aufsteigend am Ende folgen.

Ihr Script muss sauber programmiert sein: dokumentiert durch Kommentare, Überprüfen möglicher Fehler und natürlich auch sauber formatiert, damit man’s überhaupt lesen kann.

Profi-Tipp:

Leerzeichen können in Shell-Scripten Ärger machen. Es hilft in diesen Fällen, wenn man einzelne Variable in Anführungszeichen setzt, also "`${f}`" statt `${f}`. Die Schreibweise für den Zugriff auf die Variable `$f` sollte auch wegen der Leerzeichen immer mit Klammern erfolgen, d.h. `${f}`.