Übung 4 – Prozesse unter Linux und Job Control

Aufgabe 1: Skript `timer.sh`

a) Schreibe ein Skript `timer.sh`, das alle 10 Sekunden mit Hilfe des Programms `date` Datum und Uhrzeit in eine Datei schreibt (der Text soll der Datei angehangen werden), deren absoluter **Pfad als Parameter** übergeben wird.

Um auf den ersten Parameter innerhalb eines Skripts zuzugreifen, wird `$1` verwendet. z. B. gibt der Befehl

```
echo $1
```

den ersten Parameter aus, der dem Skript beim Aufruf übergeben wurde.

Mit dem Konstrukt

```
while true
do
COMMAND1
COMMAND2
done
```

werden in einer Endlosschleife die beiden Kommandos *COMMAND1* und *COMMAND2* ausgeführt. *COMMAND1* und *COMMAND2* müssen für diese Aufgabe durch die benötigten Befehl ersetzt werden.

Tipp: Benutze `sleep 10` , um die Ausführung innerhalb der `while`-Schleife zu verzögern.

b) Nachdem das Skript erstellt wurde …

1. … prüfe die Syntax von `timer.sh` wie folgt:

   ```
   bash -n timer.sh
   ```

2. … setze das „x-Flag“, um für die Datei das Recht zur Ausführung zu vergeben:

   ```
   chmod +x timer.sh
   ```

3. … starte `timer.sh` im Vordergrund:

   ```
   ./timer.sh mytimestamps
   ```

4. … und prüfe die Funktion. Wechsele dazu über die Tastenkombination [Alt]+[F2] zu einer anderen Konsole (mit [Alt]+[F1] kann man übrigens zur Standardkonsole zurückkehren). Führe dort den folgenden Befehl aus:

   ```
   tail -f mytimestamps
   ```

Tipp: Wenn man nichts mehr sieht, weil der Bildschirm durch die anderen Ausgaben überschrieben wurde, kann man zum Aufräumen ^L (**[strg]+[L]**) drücken.

c) Verständnisfragen:
- Wozu muss das „x-Flag“ gesetzt werden? Teste was passiert, wenn man versucht, ein Skript zu starten (ohne es als Parameter der `bash` zu übergeben), für das kein „x-Flag“ gesetzt ist.
- Wozu muss „`./`“ vor `timer.sh` stehen, wenn man es unmittelbar aufruft? Teste aus, was passiert, wenn man es weglässt.
- Was macht das Kommando `tail`? Was bewirkt die Option „`-f`“ dabei? Teste es aus und lies die man-Page.

Aufgabe 2: Hintergrundjobs in der Shell

a) Wechsle zu der Konsole auf der `timer.sh` gestartet wurde und stoppe das Skript durch Eingabe von ^Z ([strg]+[Z]). Was geschieht? Prüfe auf der anderen Konsole, ob weitere Ausgaben in `mytimestamps` geschehen.

b) Gebe Folgendes ein (auf der Konsole unter der `timer.sh` gestartet wurde):

```
jobs
bg
jobs
```

c) Gebe nun ein:

```
fg
```

d) Verständnisfragen und Lernfragen:
1. Wozu dienen die Kommandos `jobs`, `bg`, `fg`?
2. Finde heraus, wie man bei mehr als einem Job `fg` gezielt auf den gewünschten Job anwenden kann. Tipp: Suche nach Job Control in der man-page für `bash`.

Aufgabe 3: `timer.sh` als Hintergrundjob starten

Starte timer.sh als Hintergrundjob mit:

```
./timer.sh mytimestamps &
```

a) Ausgabe ansehen:

```
tail -f mytimestamps
```

b) Wie findet man den Job wieder?

```
jobs
```

c) Funktioniert das auch auf einer anderen Konsole? Wechsele und führe dort `jobs` aus. Weshalb ist das so?

Aufgabe 4: Befehl `ps`

a) Starte eine neue Shell mit

```
bash -norc
```

damit man den Unterschied sieht (Was macht `-norc`?)

b) Testen:

```
jobs
ps
ps -u notroot
ps -fu notroot
ps -ef
```

c) Wozu dient `ps`? Was bedeuten die einzelnen Spalten der Ausgabe? (Siehe man-page zu `ps`).

d) Zweite Shell mit ^D ([strg]+[D]) oder dem Befehl `exit` beenden.

Aufgabe 5: Befehl `kill`

Hinweis: „`kill`“ ist eigentlich ein schlechter Name … besser wäre „`send-signal`“.

a) Führe folgende Befehle aus:

```
kill
kill -l
```

b) Welchen Signalen entsprechen die folgenden Tastatureingaben?
   1. ^C ([strg]+[C])
   2. ^Z ([strg]+[Z])

c) In der Besprechung eines vorherigen Übungsblattes wurde beschrieben, dass ^D ([strg]+[D]) dafür sorgt, dass ein „Signal“ versendet wird. Handelt es sich dabei um dieselbe Art von Signal, welches vom Programm `kill` versendet werden kann?

Aufgabe 6: „Let's kill something!“

```
tail -f mytimestamps
```

a) Stelle sicher, dass `timer.sh` im Hintergrund der aktuellen Konsole rechnet. Dann ermittle die PID (Prozess-ID) von `timer.sh`. Zum Testen:

```
kill -SIGSTOP <PID von timer.sh>
kill -SIGCONT <PID von timer.sh>
```

b) Funktioniert das Gleiche auch von einer anderen Shell aus?

c) Beende `timer.sh`:

```
kill <PID von timer.sh>
```