14. Threads und `fork()`

In dieser Aufgabe testen Sie, wie es sich auswirkt, wenn Sie `pthread_create()` und `fork()` gemeinsam verwenden.

a) Schreiben Sie ein kleines Testprogramm, das zunächst mit `fork()` den aktuellen Prozess verdoppelt. Vater- und Sohnprozess erzeugen anschließend beide mit `pthread_create()` jeweils einen neuen Thread; der Vater führt darin die Funktion `vater()` aus, der Sohn die Funktion `sohn()`:

```
void *vater () {
  printf ("Neuer Thread im Vaterprozess\n");
  for (;;) ;
}

void *sohn () {
  printf ("Neuer Thread im Sohnprozess\n");
  for (;;) ;
}
```

Geben Sie auch vor und nach der Thread-Erzeugung eine kurze Meldung à la „Vater-Prozess vor pthread_create“ aus.

Starten Sie das Programm, betrachten Sie die Ausgabe und prüfen Sie (in einem anderen Terminalfenster) mit `ps -eLf`, welche Prozesse und Threads zum laufenden Programm gehören.

b) Jetzt ändern Sie die Reihenfolge: Das zweite Testprogramm erzeugt zunächst mit `pthread_create()` einen neuen Thread. Der neue Thread ruft dann `fork()` auf und beendet sich mit `return`, in der `main()`-Funktion warten Sie mit `pthread_join()` auf dieses Ereignis. Finden Sie auch hier heraus, was passiert – um die Prozesse und Threads mit `ps` beobachten zu können, bauen Sie an geeigneten Stellen wieder eine Endlosschleife `for (;;);` ein.

c) Lesen Sie den Artikel

http://www.linuxprogrammingblog.com/threads-and-fork-think-twice-before-using-them

(Link auch von der Kursseite aus erreichbar)

Der Artikel enthält eine Erklärung für das in Aufgabe **b)** beobachtete Verhalten.