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 „Threads and fork(): think twice before mixing them“

https://web.archive.org/web/20180306031630/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.

15. Parallelisieren mit Threads und Prozessen

Schreiben Sie zwei Programme, welche die Summe der ersten 100 Quadratzahlen (also

$$1^2+2^2+...+100^2$$

berechnet. Diese Aufgabe soll parallelisiert werden: im ersten Programm durch zwei Prozesse, im zweiten Programm durch zwei Threads. Dabei berechnet jeder Prozess/Thread entweder die erste oder die zweite Teilsumme:

$$1^2+2^2+...+50^2, \quad 51^2+52^2+...+100^2$$

Wenn die Teilberechnungen fertig sind, muss einer der beiden Prozesse/Threads dafür sorgen, dass beide Teilsummen zur Verfügung stehen, und die Gesamtsumme berechnen und ausgeben. Prüfen Sie, dass Sie das korrekte Ergebnis (338350) erhalten.

Ein Beispiel für die Nutzung von gemeinsamem Speicher in zwei Prozessen finden Sie auf der folgenden Seite in `shmem-beispiel.c`.

shmem-beispiel.c:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include <string.h>

int main () {
      // shared memory einrichten
      int size = 128;
      void *shmem = mmap(NULL, size,
            PROT_READ | PROT_WRITE,      // rw Zugriff
            MAP_SHARED | MAP_ANONYMOUS,  // shmem
            -1, 0);

      char *buf = (char*)shmem;
      strcpy (buf, "Originaltext");

      // Kindprozess erzeugen
      int pid = fork ();
      if (pid == 0) {
            // child
            sleep (1);
            strcpy (buf, "Erste Aenderung");
            sleep (1);
            strcpy (buf, "Zweite Aenderung");
            sleep (1);
      } else {
            // parent
            while (waitpid(pid, NULL, WNOHANG) == 0) {
                  printf ("buf = %s\n", buf);
                  usleep (50000);
            }
      }
}
```

Die Datei finden Sie auch im Moodle-Kurs und unter

http://swf.hgesser.de/bs-sp/code/shmem-beispiel.c .