Prof. Dr. Franz Schweiggert
Michael Mattes
Institut für Angewandte Informationsverarbeitung
16. Juli 2009
Blatt 12

Systemnahe Software (II)
im Sommersemester 2009

Abgabetermin: 22. Juli 2009

49 Programmdesign mit Threads (5 Punkte)

Lesen Sie das Kapitel über Threads im Skript, die Manpage von `pthread_create` sowie den Abschnitt "Designing Threaded Programs" der Webseite
`https://computing.llnl.gov/tutorials/pthreads/#Designing`
durch. Erklären Sie dann Ihrem Tutor / Ihrer Tutorin in eigenen Worten folgende Dinge:

- Wie lauten die wichtigsten Funktionen zum Umgang mit Threads?
- Wie kann man einen Thread mit Nicht-Standard-Attributen erstellen?
- Kann ein Thread auf den nächsten freiwerdenden Thread warten? Wenn ja, wie?
- Wofür sind parallelisierte Programme besonders geeignet?
- Woran erkennt man, ob sich ein Programm gut parallelisieren lässt?
- Welche Software-Designmuster zur Verwendung von Threads gibt es?
- Wie funktioniert eines dieser Muster im Detail?
- Was ist der Unterschied zwischen einem statischen und einem dynamischen Workerpool beim Manager/Worker-Muster? *Diese Frage wird nicht auf der angegebenen Webseite beantwortet, eine Suche nach "Thread pool pattern" hilft aber vielleicht.*

Hinweis: Auch der Rest der angegebenen Webseite liefert viele Informationen zum Umgang mit Threads und ist eine gute Zusatzlektüre.

50 Sichere Kontobewegungen mit Mutual Exclusion (15 Punkte)

Nach 11 Blättern und 49 Aufgaben sind wir nun am Ende angelangt. Da die Vorlesung vor allem Software zum Thema hat, sollen Sie zum Abschluss noch ein letztes Stück Software schreiben.

Erstellen Sie ein Programm `kontostand.c`, welches Kontobewegungen simuliert. Hierzu soll zunächst der Kontostand (in Euro, als Ganzzahl) von der Standardeingabe gelesen werden. Danach sollen die Kontobewegungen simuliert werden, die über die Kommandozeile gegeben sind. Jede Kontobewegung soll in einem eigenen Thread abgearbeitet werden. Der Gesamtkontostand soll eine von allen Threads geteilte Variable sein. Sorgen Sie mit Hilfe eines Mutex dafür, dass der Kontostand am Ende korrekt ist!

Ein Beispiel für die Ausgabe folgt:

```
$ echo "1000" | ./kontostand 100 150 -200
Hauptthread: Der Anfangs-Kontostand ist 1000 Euro.
Thread 2: Aktueller Kontostand: 1000 Euro. Veraendere Stand um 100 Euro.
Thread 3: Aktueller Kontostand: 1000 Euro. Veraendere Stand um 150 Euro.
Thread 2: Neuer Kontostand: 1100 Euro.
Thread 4: Aktueller Kontostand: 1100 Euro. Veraendere Stand um -200 Euro.
Thread 3: Neuer Kontostand: 1150 Euro.
Thread 4: Neuer Kontostand: 900 Euro.
Hauptthread: Der End-Kontostand ist 900 Euro.
```

In obigem Beispiel sieht man, was ohne Synchronisierung passieren kann. Die Änderung von Thread 3 ging komplett verloren ("Lost update").

Viel Erfolg!