Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Betriebssysteme Praktikum 2

Entwurfsbeschreibung

Florian Heuer & Desirée Pehmöller

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Kurzbeschreibung

Im Praktikum 2 der Veranstaltung “Betriebssysteme” soll eine Simulation trainierender Philosophen in einer „Muckibude“ mit Hilfe von Threads, Semaphoren, Conditionvars und Mutexen (Monitorkonzept) realisiert werden. Dabei soll der Umgang mit Threads, insbesondere dessen Synchronisierung mit Werkzeugen des Betriebssystems erlernt werden.

2 Entwurf

2.1 Beschreibung

Der Entwurf für das Programm „**MuscleFactory**“ sieht einen ähnlichen Ansatz wie in einer objektorientierten Sprache vor. In der eingesetzten Sprache C ist dies nicht ohne Weiteres möglich. Eine strukturierte Weise mit gekapselten Zuständigkeiten wird angestrebt und so wird der Code über mehrere dedizierte Sourcefiles aufgeteilt auf diese in 2.1.2 eingegangen wird. Zunächst wird die Umgebung beschrieben in der das Programm entwickelt wird.

2.1.1 Umgebung

Das Programm wird in Sprache C entwickelt. Als Compiler kommt „GCC“ zum Einsatz. Das Betriebssystem Open SUSE in Version 12.3 wird zum kompilieren und ausführen des Codes benutzt. Als Editor wird Sublime 2 verwendet. Zudem wird zur Erleichterung des Kompiliervorgangs ein Makefile genutzt.

2.1.2 Konzept

Wie oben erwähnt wird das Programm in verschiedene Files gekapselt. Diese werden unterhalb aufgeführt und relevante Details näher erläutert.

1. **Global.h**
   Hier werden benötigte globale Variablen, Funktionen und Strukturen definiert.

   Ein **Philosoph** wird als Struktur und eigener Typ abgebildet und hält dabei die Attribute für eine Thread ID, zu trainierendes Gewicht, Name, Modus (ob er blockiert, normal oder beendet ist), Status (ob er sich Gewichte holt, trainiert,

Gewichte zurücklegt oder ruht), geholte Gewichte (Struktur mit 3 Integern repräsentativ für 2, 3, und 5kg) und eine Semaphore für die Blockierung des Threads.

Globale Variablen: Ein Array welches alle Philosophen (oben beschriebene Struktur) enthält. Ein Gewichtedepot, welches alle Gewichte der Muckibude enthält. Eine Conditionvariable und ein Mutex zur Threadkoordinierung. Char Array zum speichern eines gelesenen Kommandos.

Funktionsprototypen: (get__status) Funktion zur Ausgabe des Zustandes der gesamten Muckibude. (getThreadID) Filterfunktion zum Auslesen der Thread ID aus einem übergebenen Komanndo.

2. Main.c

Implementationen der Funktionen get_status und getThreadID.
In der Main Methode werden alle Philosophen und Threads erzeugt. Ein Thread erhält dabei genau einen Pointer auf einen Philosohpen.
In einer While-Schleife werden danach auf Komandos mit fgets eingelesen und in dem globalen Char-Array gespeichert. Wird das Kommando „q“ eingelesen wird die Schleife beendet und auf alle Threads gewartet bis sich diese ebenfalls beendet haben. Das Programm endet. Das Kommando <id>u sorgt für die Freigabe eines blockierten Threads.

3. Philosoph.c

Hier wird die Threadfunktion implementiert, welche in der Main gestartet wird. Als parameter bekommt die einen Pointer auf eine Philosophenstruktur. Der Traningszyklus (Gewichte holen, trainieren, Gewichte zurücklegen und ruhen) eines Philsophen wird in der Funktion implementiert. Dabei ist zu beachten das hier der Eintritt in den kritischen Bereich mit Zugriff auf das Gewichtedepot erfolgt. Der kritische Bereich wird hier mit einer Conditionvariable und einem Mutex geschützt (Monitorkonzept).

Des Weiteren werden 3 Hilfsmethoden implementiert. Workout(), rest(), und read_input() welche in der Threadfunktion aufgerufen werden. In den Funktionen rest() und work() werden Befehle ausgelesen welche für eine Beendigung des Trainingszyklus und für die Blockierung des Threads sorgen.

4. MuscleFactory.c

In dieser Sourcedatei werden die Funktionen GET_WEIGHTS und PUT_WEIGHTS implementiert, welche für den Zugriff auf das Gewichtedepot der Muckibude

zuständig sind. GET_WEIGHTS erhält dabei als einzigen Parameter den Philosopen, dieser hält das Trainingsgewicht welches vom Philosophen geholt werden möchte. Der Algorithmus entscheidet nun welche Kombination an Gewichten an den Philosophen gegeben wird bzw. ob dies möglich ist. Die Rückgabe ist 0 (Gewichte konnten nicht geholt werden, weil keine Kombination zum gewünschten Gewicht realisierbar ist.) oder 1 (Gewichtekombination wird an den Philosoph übergeben).

2.2 Repository

Der komplette Sourcecode des Programms "**MuscleFactory**" ist unter folgendem Link auf GitHub zu finden.

https://github.com/FlowwX/muscleFactory