5 Übungsaufgabe #5: Fäden und Fadenwechsel

In den vorigen Aufgaben wurde eine Unterbrechungsbehandlung implementiert, mit der die aus der Betriebssystemvorlesung bekannten Hardwareunterbrechungen durch POSIX Signale emuliert werden können.

5.1 Kontrollfäden und Kontextwechsel

Um mehr als einen Aktivitätsträger auf einem Prozessor laufen zu lassen, muss das Betriebsmittel CPU virtualisiert werden. Hierzu wurde bereits eine Coroutinenimplementierung in den Übungen zur Lehrveranstaltung Betriebssysteme für Einprozessorsysteme vorgestellt und implementiert. Zur Erinnerung: Dort wurde streng zwischen der Umschaltung (engl. *dispatching*) und Fadeneinplanung (engl. *scheduling*) unterschieden. Für Mehrkernsysteme können die Techniken zur Fadenumschaltung mit wenigen Anpassungen übernommen werden.

5.2 Multiprozessor Scheduling

In dieser Aufgabe wird ein prototypischer, *Round-Robin*-artiger Scheduler implementiert. Zur Vereinfachung werden Prozesse in dieser Aufgabe nicht von einer CPU auf eine andere migriert, sondern werden auf einer zum Erstellungszeitpunkt festgelegten CPU gehalten. Dies wird in den folgenden Aufgaben erweitert werden. Die eigentliche Umschaltung erfolgt präemptiv aus dem Timerinterrupt heraus.

5.3 Primitiven zur Ausgabe

Damit die Implementierung besser getestet werden kann, soll die Ein/Ausgabebibliothek aus Aufgabe 2 so erweitert werden, dass der Benutzer die Position am Bildschirm angeben kann, an der nachfolgende Ausgaben gemacht werden. Zur Implementierung könnt Ihr auf ANSI-Escape-Sequenzen zurückgreifen.
Die Testanwendung soll in Abhängigkeit der CPU und der Fadenkennung an einer bestimmten Position eine wechselnde Ausgabe machen. Dadurch soll der Fortschritt z.B. einer Berechnung simuliert werden.

Aufgaben:

→ Implementieren Sie eine Koroutine und einen passenden Dispatcher zur Umschaltung. Die Koroutinen sollen dabei zur Laufzeit erstellt und zerstört werden können. Der Kontext soll im Gegensatz zur Betriebssystemvorlesung nicht in einen Prozesskontrollblock, sondern komplett auf den Stack gesichert werden.

Der Timerinterrupt sollt über alle virtuellen Prozessoren verteilt werden, so dass man nur eine einzige Zeitgeberquelle benötigt.

→ Implementieren Sie einen einfachen, präemptiven *Round-Robin*-artigen Scheduler mit prozessorlokaler Bereitliste. Eine Migration von Fäden auf eine andere CPU ist in dieser Aufgabe (noch) nicht gefordert.

→ Schreiben Sie ein Testprogramm, welches den Fadenwechsel auf allen Prozessoren demonstriert. Auf jeder CPU sollen dabei (mindestens) zwei Testprogramme gestartet werden.

→ Erweitert eure Ausgabestreamklasse um einen Operator zum Setzen der Position des Cursors.

Hinweise:

- Es bietet sich an die Position in einer Klasse `gotoxy` zu kapseln, die dann von einem speziell überladenen Operator `<<` euerer Streamklasse ausgewertet wird, so dass Ausgaben wie im untenstehenden Beispielcode funktionieren:

```
int main() {
    cout << gotoxy(10,11) << "Hello Ansi" << endl;
}
```

5.4 Abgabe: am 13.07.2011