Safety-Critical Systems 3:
Methods & Tools

WiSe 07/08

AG Betriebssysteme, FB 3
Jan Peleska
Christof Efkemann

ausgegeben: 15.11.2007
Abgabe: 29.11.2007

Aufgabe 3

Implementieren mit Z: Der Fahrstuhl

Implementiere die Fahrstuhlsteuerung, die wir in Z spezifiziert haben, unter Linux in C++.

Beachte dabei die in der Vorlesung genannten Punkte:

1. Scheduling der Operationen
2. Interface-Realisierung
3. Zeitbedingungen
4. Abbildung mathematischer Datentypen
5. Umsetzung impliziter Operationsspezifikationen in Algorithmen

Die eigentliche Fahrstuhlsteuerung soll als ein Linux-Prozess realisiert werden. Die Hardware soll durch parallel laufende Linux-Prozesse simuliert werden. Alle Prozesse sollen über gemeinsamen Speicher kommunizieren.

Die Simulation des Knopfcontrollers soll in einem eigenen X-Terminal laufen und die Eingabe von Stockwerkswünschen erlauben.

Ein weiterer Prozess soll die Simulation des physikalischen Modells übernehmen. Er soll den physikalischen Zustand des Motors simulieren und zu geeigneten Zeitpunkten ein neues Stockwerk an den Sensor übermitteln (alles ebenfalls über gemeinsamen Speicher). Der Prozess soll ebenfalls in einem eigenen X-Terminal laufen und die jeweiligen Ausgabewerte dort ausdrucken.

Lösungshinweise:

- Beispielcode für die Kommunikation über gemeinsamen Speicher findet Ihr z.B. auf den Webseiten der Vorlesung „Betriebssysteme 1“
  `www.informatik.uni-bremen.de/agbs/lehre/ss03/bs1/`
  in der Literaturliste am Ende von Session 1: „Consumer“ und „Producer“.
- Verwendet folgende Konstanten und folgende Datenstruktur für den gemeinsamen Speicher:

```
#ifndef _SCS3_ELEV_H_
#define _SCS3_ELEV_H_

#define TOP_FLOOR 8
#define GROUND_FLOOR 0
```

```
#define NO_FLOOR -1

#define UP 1
#define DOWN 2
#define STOPPED 3

#define FLOOR_Q_SIZE (TOP_FLOOR - GROUND_FLOOR + 2)

typedef struct {
    /* controller input: floor sensor */
    int sensor;
    /* controller input: ring buffer of pending floor requests */
    int targetFloorQ[FLOOR_Q_SIZE];
    /* index of first pending floor in targetFloorQ */
    int targetFloorFirst;
    /* index of last pending floor in targetFloorQ */
    int targetFloorLast;
    /* controller output: state of motor */
    int motorState;
} elev_shm_t;

#endif
```

- Beachtet, daß wir zwei verschiedene Arten von Eingaben haben: Der Stockwerksensor kann beliebig oft gelesen werden, ein Knopfdruck ist dagegen nur einmal verfügbar.
- Eine Zeitverzögerung im physikalischen Modell erreicht Ihr mit der Betriebssystemfunktion:
  `unsigned int sleep(unsigned int seconds);`
- Als periodische Timer können z. B. benutzt werden:
  - `setitimer(ITIMER_REAL, ...)`
  - `ioctl RTC_PIE_ON` (siehe `man rtc`)
- Quellcode bitte hinreichend kommentieren!

Abgabe per E-Mail an `chref@tzi.de` (als PDF-Datei und Source Code).