

# Aufgabenblatt 3

## Echtzeitsysteme

Institut: Berliner Hochschule für Technik  
Dozent: Prof. Dr. Christian Forler  
Url: <https://lms.bht-berlin.de/>  
Email: [cforler\(at\)bht-berlin.de](mailto:cforler@bht-berlin.de)  
SoSe 2022

### **Aufgabe 1 3-Zustands-Prozessmodell Simulator (8 Punkte)**

Upgraden Sie Ihren 2-Zustands-Prozessmodell Simulator (Aufgabenblatt 1) zu einem 3-Zustands-Prozessmodell Simulator. Gehen Sie dazu wie folgt vor

1. Erweitern Sie den Typ `enum state` um den Wert `Blocked`. Fügen Sie eine Funktion hinzu, welche die neunen Zustandsänderung analog zu `p_switch_state()`, für einen Prozess, durchführt.
2. Erweitern Sie den Verbund `struct pctx` um den Member `struct queue *qblocked`.
3. Passen Sie ihre `print()`-Funktionen an.
4. Richten Sie einen Signalhandler für das Signal `SIGUSR1` ein. Bei dem Eintreffen des Signals `SIGUSR1` soll der laufende Prozess der Warteschlange `qblocked` hinzugefügt werden.
5. Richten Sie einen Signalhandler für das Signal `SIGUSR2` ein. Bei dem Eintreffen des Signals `SIGUSR2` soll ein Prozess aus der Warteschlange `qblocked` entfernt und anschließend der Warteschlange `qready` hinzugefügt werden.
6. Erstellen Sie eine Demoanwendung welches die Zustandsübergänge des 3-Zustands-Prozessmodells mit 10 Prozessen simuliert. Nach der Initialisierung soll der Prozess-Kontext durch eine neue Zustandsänderung modifiziert und ausgegeben werden.

### **Aufgabe 2 Signalmengenfunktionen (4 Punkte)**

Erstellen sie eine C-Bibliothek, welche Bitmasken verwendet. Die Bibliothek soll über die folgenden Funktionen verfügen: `my_sigemptyset()`, `my_sigfillset()`, `my_sigaddset()`, `my_sigdelset()` und `my_sigismember()`. Das Verhalten der Funktionen soll denen ohne den Prefix `my_` entsprechen. Beispielsweise soll das Verhalten von `my_sigemptyset()` analog zu dem von `sigemptyset()` sein. Ihre Lösung soll mit Bitmasken arbeiten bei denen jedes Signal durch ein einzelnes Bit repräsentiert wird. Schreiben Sie eine Testanwendung die ihre Bibliothek ausgiebig testet.

*Hinweise:*

- *Es ist ratsam sich einen eigenen Datentyp `my_sigset_t` zu definieren.*
- *Sie können davon ausgehen das es nicht mehr als 32-Signale gibt.*

### **Aufgabe 3 Auf die Kinder warten**

**(4 Punkte)**

Schreiben Sie ein Programm `childwait` welches  $k$  Kindprozesse generiert. Die Anzahl  $k$  soll als Kommandozeilenparameter übergeben werden. Bei dem Start eines Kindprozesses soll der Elternprozess eine globale Variable  $n$  um 1 inkrementieren. Bei Beendigung eines Kindprozesses, was dem Elternprozess mit dem Signal `SIGCHLD` mitgeteilt wird, soll er die globale Variable  $n$  um 1 dekrementiert werden. Richten Sie dazu einen **Signalhandler** ein. Wenn  $n == 0$  gilt, soll sich der Elternprozess beenden.

*(Hinweis: Alle Ausgaben sollen von dem Elternprozess generiert werden.)*

```
./childwait 2
Child 3534: started (n=1)
Child 3535: started (n=2)
Parent 3533: sleep(2)
Child 3535: terminated (n=1)
Parent 3533: sleep(2)
Child 3534: terminated (n=0)
Parent 3533: terminated
```