Aufgabenblatt 3
Echtzeitsysteme (SoSe 2018)

Institut: Beuth Hochschule für Technik Berlin
Dozent: Prof. Dr. Christian Forler
Url: `https://lms.beuth-hochschule.de/`
Email: `cforler(at)beuth-hochschule.de`

Aufgabe 1 (8 Punkte) 3-Zustands-Prozessmodell Simulator
Upgraden Sie Ihren 2-Zustands-Prozessmodell Simulator (Aufgabenblatt 1) zu einem 3-Zustands-Prozessmodell Simulator. Gehen Sie dazu wie folgt for

1. Erweitern Sie den Typ `enum state` um den Wert `Blocked`
2. Erweitern Sie den Verbund `struct pctx` um den Member `struct queue *qblocked`
3. Passen Sie ihre `print()`-Funktion.
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 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 (4 Punkte) Signalmengenfunktionen
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. Gehen Sie wie folgt vor.

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 (4 Punkte) Auf die Kinder warten
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.

```
./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
```

Aufgabe 4 (4 Punkte) Signal vs. Realtime-Signal
Angenommen zwei Signale hängen. Bei dem einen handelt es sich um ein reguläres Signal und bei dem anderen um ein Echtzeitsignal.

In dem POSIX Standard ist nicht spezifiziert welches der beiden Signale zuerst zugestellt wird. Schreiben Sie eine Programm das herausfindet wie sich der Linux-Kernel in diesem Fall verhält.

Aufgabe 5 (8 Punkte) Parallele Quersummenberechnung I
Erstellen Sie sich mit `https://www.random.org/integers/` eine Liste von 1000 Zufallszahlen zwischen 1 und 1000 und speichern sie diese unter `number.txt` ab

a) Schreiben Sie eine Methode `int sum_of_digits(char *num)` welche die Quersumme einer Zahl `num` berechnet. Beispielsweise ist die Quersumme von `"1234"` gleich 10. Handelt es sich bei `num` nicht um eine Zahl gibt die Funktion `-1` zurück. Beispiel: `sum_of_digits("not a number") == -1`.

b) Schreiben Sie ein Programm welche die Datei `number.txt` als Kommandozeilenparameter einliest, von jeder Zahl die Quersumme berechnet und am Ende die Summe aller Quersummen ausgibt. Das Programm soll die Funktion `int sum_of_digits()` aus dem Aufgabenteil a) verwenden.

c) Schreiben Sie eine alternative Lösung für Aufgabenteil b) welche mittels `fork()` n Prozesse (z.B. 5) startet. Die generierten n Kindprozesse sollen parallel die Quersummen der einzelnen Zahlen berechnen. Die Arbeit soll unter den Kindern fair aufgeteilt werden. Der Elternprozess soll am Ende die Summe aller Quersummen ausgeben.

   Die Zwischenergebnisse sollen als Payload eines Echtzeitsignal übertragen werden. Richten Sie dazu einen entsprechenden Signalhandler ein.