Echtzeitsysteme: 2. Aufgabenblatt

Sommersemester 2020
Dr.-Ing. René Tschirley

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

2-Zustands-Prozessmodell

letzte Änderung: 6. Mai 2020

Erstellen Sie ein C-Programm, welches die Änderung von Prozesszuständen innerhalb des 2-Zustands-Prozessmodells simuliert.

1. Aufgabe: Prozess

Abgabedatei(en): `process.h`, `process.c`

Erstellen Sie eine Datei `process.h` mit den folgenden Inhalten

- Eine Enum `enum state` welches die beiden Werte `READY` und `RUNNING` annehmen kann.
- Ein Verbund `struct process` welcher über die Elemente `uint32_t p_id` und `enum state p_state` verfügt.
- Den Funktionsprototypen `void p_switch_state(struct process *p)`. Die Implementierung soll den Zustand eines Prozesses von `READY` auf `RUNNING` bzw. von `RUNNING` auf `READY` verändern.
- Den Funktionsprototypen `void p_print(struct process *p)`. Die Implementierung soll den übergeben Prozesses `p` in menschenlesbarer Form ausgeben.

Erstellen Sie ein Datei `process.c` welche die Funktionsprototypen aus `process.h` implementiert.

Schreiben Sie ein Programm, welches die implementierten Funktionen prüft.

2. Aufgabe: Warteschlange

Abgabedatei(en): `queue.h`, `queue.c`

Implementieren Sie die Warteschlangen (`queue`) als verkette Listen. Gehen Sie dazu wie folgt vor:

Erstellen Sie eine Datei `queue.h` mit den folgenden Inhalten:

- Ein Verbund `struct q_node` mit den folgenden Elementen:
  `struct q_node *next` und `struct process *p`.
- Ein Verbund `struct queue` mit den folgenden Elementen:
  `struct q_node *start` und `struct q_node *end`.
- Den Funktionsprototypen `void q_add(struct queue *q, struct process *p)`. Bei dieser Funktion soll der Prozess `p` der Warteschlange `q` hinzugefügt werden. Der Prozess wird damit das letzte Element in der Warteschlange.
- Den Funktionsprototypen `struct process *q_remove(struct queue *q)`. Die Implementierung soll das erste Element der Warteschlange `q` entfernen und zurückgeben.
- Den Funktionsprototypen `void q_print(struct queue *q)`. Die Implementierung soll die übergebene Warteschlange `q` in menschenlesbarer Form ausgeben.

Erstellen Sie ein Datei `queue.c` welche die Funktionsprototypen aus `queue.h` implementiert.

Schreiben Sie ein Programm `queuedemo.c` welches Ihre Implementation prüft und ein Makefile, welches Ihr Testprogramm baut.

3. Aufgabe: Process Model

Abgabedatei(en): `processmodel.h`, `processmodel.c`

Implementieren sie das 2-Zustands-Prozessmodell Simulator indem Sie wie folgt vorgehen:

Erstellen Sie eine Headerdatei `processmodel.h` mit den folgenden Komponenten:

- Ein Verbund `struct pctx` mit den folgenden Elementen:
  `struct queue *qready` und `struct process *running` verfügt.
- Einen Funktionsprototypen `void print(struct pctx *ctx)`. Die Implementierung soll den Kontext `ctx` in menschenlesbarer Form ausgeben.
- Einen Funktionsprototypen `void step(struct pctx *ctx)`. Die Implementierung soll den momentan laufenden Prozess `running` in die Warteschlange `qready` hinzufügen und der erste Prozess in `qready` wird zum neuen laufenden Prozess.

Erstellen Sie eine Demoanwendung welche die Zustandsübergänge des 2-Zustands-Prozessmodells mit 10 Prozessen simuliert. Nach der Initialisierung soll der Prozess-Kontext durch eine neue Zustandsänderung modifiziert und ausgegeben werden.