

# Übungszettel 1

## Hinweise

Die Abgabe erfolgt als Ausdruck am Ende der Vorlesung und als E-Mail an *kirsten@tzi.de*. **Auf jeden Fall** sollten alle C-Dateien auch in elektronischer Form (als E-Mail-Attachment) abgegeben werden. Zur vollständigen Lösung der Aufgabe gehören Programm, Test und Dokumentation (in Latex). Der Betreff der E-Mail sollte folgendes Aussehen haben:

**BS1 Abgabe x Gruppe y.**

Bitte immer die Namen aller Gruppenmitglieder und die Gruppennummer angeben!

## Aufgabe 1: Ringpuffer für Elemente mit variabler Länge

Programmieren Sie eine C-Bibliothek, welche einen Ringpuffer für Telegramme variabler Länge bereitstellt. Diese kann z.B. für die Punkt-zu-Punkt Kommunikation zwischen verschiedenen Threads eingesetzt werden. Testen Sie die Bibliothek.

Berücksichtigen Sie dabei auch, dass die Bibliothek auf verschiedenen Rechnerarchitekturen laufen soll. Auf SPARC-Architekturen können Integer-Werte (4 Bytes) nur an den entsprechenden Wort-Grenzen abgelegt werden.

Die Bibliothek soll folgende Funktionen enthalten:

### Erzeugen eines FIFO Ringpuffers

Erzeugt einen FIFO Ringpuffer der Gesamtgröße *buffer\_size* Bytes, der für die Kommunikation von Thread *sourceThreadId* nach Thread *targetThreadId* zu verwenden ist. Der Puffer wird dynamisch erzeugt. Die Verwaltungsinformationen (Zeiger auf den Pufferanfang und die Puffergröße, Quell- und Ziel-Thread Id, Lese- und Schreibindizes) werden in einer ebenfalls dynamisch zu allozierenden Struktur *struct Rb\_handle\_t* registriert und an den Aufrufer zurückgegeben.

```
struct Rb_handle_t *initRb(int targetThreadId,  
                           int sourceThreadId,  
                           size_t buffer_size);
```

### Element in den Puffer schreiben

Schreibt ein neues Element *item* der Länge *item\_size* in den Ringpuffer. Der Befehl schlägt fehl, wenn eine der folgenden Bedingungen erfüllt ist:

- *targetThreadId/sourceThreadId* passen nicht zum *handle*
- es ist nicht mehr genügend Platz im Puffer, um den *item* der Länge *item\_size* Bytes im Ringpuffer unterzubringen

```
int writeRb(struct Rb_handle_t *handle,
           int targetThreadId,
           int sourceThreadId,
           const void *item,
           size_t itemsize);
```

## Element aus dem Puffer lesen

Das Lesen des ersten Elements aus dem Ringpuffer erfolgt analog. Beim Aufruf der Funktion trägt man in *itemsize* die Länge des vom Anwendungsprogramm zur Verfügung gestellten Puffers ein. Bei Rückkehr aus der Funktion ist dort die Länge der tatsächlich nach *item* kopierten Daten enthalten.

Der Rückgabewert ist

- 1: falls erfolgreich und vollständig nach *item* kopiert wurde
- 0: falls nichts im Puffer war
- -1: falls nicht das ganze Telegramm in *item* passte. In diesem Fall wird *item* komplett gefüllt und der Rest kann mit einem nachfolgenden Aufruf abgeholt werden. Er geht also nicht verloren.
- -2: bei Fehlern analog zu *writeRb()*

```
int readRb(struct Rb_handle_t *handle,
           int targetThreadId,
           int sourceThreadId,
           void *item,
           size_t *itemsize);
```

## Ringpuffer löschen

Dealloziert alle zum Ringpuffer gehörigen Puffer.

```
int freeRb(struct Rb_handle_t *handle);
```