Thread API & Locks

Prof. Dr.-Ing. Andreas Heil

v1.0.2

Thread API

Fragestellung: Welche Schnittstellen muss das Betriebssystem bereitstellen, um Threads zu erstellen und zu kontrollieren?

- Am Beispiel von POSIX (Portable Operating System Interface) Systemen:
  - Ein Thread erzeugen
  - Auf einen Thread warten
  - Locks = Ausschluss aus kritischem Abschnitt
  - Condition Variable (dt. Monitor = Synchronisationsmechanismus)

Ein Thread erzeugen

4 Argumente erforderlich: `thread`, `attr`, `start routine`, `arg`

```
#include <pthread.h>
int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine)(void*),
                   void *arg);
```

thread und attr

- `thread`
  - Pointer auf Struktur vom Typ `pthread_t`, wird genutzt um mit dem Thread zu interagieren
- `attr`
  - Spezifiziert die Attribute des Threads (Größe d. Stacks, Scheduling Priorität etc.)

start_routine und arg

- `start_routine`
  - Function Pointer auf Routine, die vom Thread ausgeführt werden soll
- `arg`
  - Argument, das an den Anfang des Threads übergeben wird

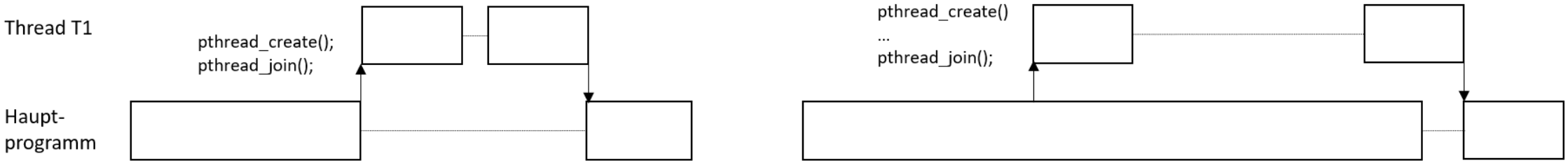
Auf einen Thread warten

```
int pthread_join(pthread_t thread, void **value_ptr);
```

- `pthread_t`
  - Spezifiziert den Thread, auf den gestartet wird
- `value_ptr`
  - Zeiger auf den Rückgabewert der Routine

To thread or not to thread

- Einen Thread mittels `pthread_create` zu erzeugen und direkt danach mittels `pthread_join` auf dessen Beendigung zu warten ist natürlich nicht sonderlich sinnvoll
- In diesem Fall ist ein einfacher Prozeduraufruf einfacher.
- Wenn der Thread jedoch gestartet wird, und erst im späteren Verlauf des Programms auf seine Beendigung gewartet wird, macht es wieder Sinn



Locks

- Funktionen für den gegenseitigen Ausschluss
- **Mutex:** Mutual Exclusion Object

```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock(&lock);
x = x + 1; // critical section
pthread_mutex_unlock(&lock);
```

Wie funktioniert ein Mutex?

- Kritischer Abschnitt kann nur betreten werden, wenn man (der Code) in Besitz des Mutex ist
- Beispiel hier: `pthread_mutex_loc` setzt das Mutex, und kein anderer Thread (mit dem gleichen Code) kann diesen Code-Abschnitt betreten, bevor `pthread_mutex_unlock` aufgerufen wurde
- Wie wird das erreicht?
  - Ist der Lock gesetzt, kehrt die Routine `pthread_mutex_lock` erst zurück, nachdem das Mutex freigegeben wurde

Condition Variables

Verständigung zwischen Threads

- z.B. ein Thread wartet auf etwas von einem anderen Thread
- `pthread_cond_wait` lässt den Thread warten, bis er das entsprechende Signal bekommt, dass er weitermachen kann

```
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_cond_signal(pthread_cond_t *cond);
```

Referenzen