Abgabe zur 1. Praktikumsaufgabe im Fach „Embedded Computing“

erstellt von

Markus Schmidt
Maximilian Gaul

Unterschied DETACHED und JOINABLE in *pthread_create* bzw. *pthread_attr_setdetachstate*:

Bei Detached wird der Threadkontext gleich nach verlassen durch den Thread gelöscht.

Bei Joinable wird dieser nicht durch den Thread selbst bei Beendigung gelöscht, sondern erst durch die Funktion pthread_join. Damit werden die Ressourcen erst nach Aufruf von pthread_join freigegeben. Werden in einem Programm sehr viele Threads mit „Joinable“ erstellt, und der Aufruf von pthread_join vergessen, kann das zu einem Speicherproblem führen (Ressourcen werden nicht freigegeben).

Durch die Einbindung über eine Library kann der Thread nicht erkennen, dass jemand auf ihn wartet. Somit ist wichtig, dass man diesem mitteilt, dass jemand auf ihn wartet und er seinen Threadkontext nicht löscht.

Über den Threadkontext kann der Aufrufer von pthread_join prüfen, ob der Thread:

- Existiert hat, aber schon beendet wurde (Kontext noch vorhanden, wird erst durch pthread_join gelöscht)
- Existiert, aber noch läuft
- Noch nie existiert hat. (Kein Kontext vorhanden)

Verarbeitung des Rückgabewertes aus dem jeweiligen Thread:

Der Thread erhält die eigene Task-ID vom Typ *pthread_t* mit dem Aufruf von *pthread_self()* und gibt diese als Returnwert zurück, in dem er der Funktion *pthread_exit()* als *void** übergeben wird (die Funktion erwartet einen *void**):

Code: *pthread_exit((void*)pthread_self());*

pthread_join() speichert den Rückgabewert (*void**) des Threads in einen Speicherbereich, die wir als Referenz auf eine Variable vom Typ *pthread_t* (derselbe Rückgabewert wie *pthread_self()*) übergeben:

Code: *pthread_t p1IdAfter;*
Code: *pthread_join(p1Thread, (void**)&p1IdAfter);*

Durch des gleichen Typs *pthread_t* wird garantiert, dass nur der Speicherbereich reserviert wird, der auch tatsächlich zum Schreiben des Rückgabewertes benötigt wird. Daher ist die Handhabung als *void** möglich.