Beispiele zu Kapitel 7: Programmierung planbarer Systeme

Aus: Alan Burns, Andy Wellings: *Real-Time Systems and Programming Languages. Ada, Real-Time Java and C/Real-Time POSIX*. Addison Wesley, 2009.(Kapitel 12, Beispiel 12.2 und 12.3)

Die C/Real-Time POSIX Schnittstelle für Real-Time Scheduling

Konstanten für Scheduling-Strategien und -Optionen:

```
#define SCHED_FIFO ...      /* preemptive priority scheduling */
#define SCHED_RR ...        /* preemptive priority with quantum */
#define SCHED_SPORADIC ... /* sporadic server */
#define SCHED_OTHER ...     /* implementation-defined scheduler */
#define PTHREAD_SCOPE_SYSTEM ... /* system-wide contention */
#define PTHREAD_SCOPE_PROCESS ...  /* local contention */
#define PTHREAD_PRIO_NONE ... /* no priority inheritance */
#define PTHREAD_PRIO_INHERIT ... /* basic priority inheritance */
#define PTHREAD_PRIO_PROTECT ... /* ICPP */
```

Datentypen für Scheduling-Parameter:

```
typedef ... pid_t;

struct sched_param {
  ...
  int sched_priority; /* used for SCHED_FIFO and SCHED_RR */
  ... };
```

Bearbeiten der Scheduling-Parameter für Prozesse:

```
int sched_setparam(pid_t pid, const struct sched_param *param);
int sched_getparam(pid_t pid, struct sched_param *param);
/* set/get the scheduling parameters of process pid */

int sched_setscheduler(pid_t pid, int policy,
                       const struct sched_param *param);
int sched_getscheduler(pid_t pid);
/* set/get the scheduling policy of process pid */

int sched_yield(void);
/* causes the current thread/process to be placed at the back */
/* of the run queue */

int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
/* returns the max/minimum priority for the policy specified */

int sched_rr_get_interval(pid_t pid, struct timespec *t);
/* if pid /= 0, the time quantum for the calling process/thread is
   set in the structure referenced by t
   if pid = 0, the calling process/threads time quantum is set
   in the structure pointed to by t
 */
```

Bearbeitung der Scheduling-Parameter für Threads (über die Thread-Attribute, vgl. Beispiele zu Kapitel 2):

```
int pthread_attr_setscope(pthread_attr_t *attr,
                          int contentionscope);
int pthread_attr_getscope(const pthread_attr_t *attr,
                          int *contentionscope);
/* set/get the contention scope attribute for a */
/* thread attribute object */

int pthread_attr_setschedpolicy(pthread_attr_t *attr,
                                int policy);
int pthread_attr_getschedpolicy(const pthread_attr_t *attr,
                                int *policy);
/* set/get the scheduling policy attribute for a */
/* thread attribute object */

int pthread_attr_setschedparam(pthread_attr_t *attr,
                                const struct sched_param *param);
int pthread_attr_getschedparam(const pthread_attr_t *attr,
                                struct sched_param *param);
/* set/get the scheduling parameter attribute for a */
/* thread attribute object */
```

Bearbeitung der Scheduling-Parameter für Mutexe (Prioritätsvererbung, Prioritätsobergrenzen):

```
int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,
                                  int protocol);
int pthread_mutexattr_getprotocol(pthread_mutexattr_t *attr,
                                  int *protocol);
/* set/get the priority inheritance protocol */

int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,
                                     int prioceiling);
int pthread_mutexattr_getprioceiling(pthread_mutexattr_t *attr,
                                     int *prioceiling);
/* set/get the priority ceiling */

/* All the above integer functions return 0 if successful */
```