Echtzeitbetriebssysteme — Übung

Oliver Jack

Ernst-Abbe-Hochschule Jena
Fachbereich Elektrotechnik und Informationstechnik

Sommersemester 2025

Ernst-Abbe-Hochschule Jena
University of Applied Sciences

Praktikum 6: Semaphores

Semaphore

- VxWorks bietet für den exklusiven Zugriff auf Ressourcen Semaphore an.
- Semaphore in VxWorks sind hoch optimiert und erlauben extrem schnelle Intertask-Kommunikation.
- Semaphore sind die bevorzugten Mittel für den wechselseitigen Ausschluss (mutual exclusion) und die Tasksynchronisation.

Beschreibung

Es gibt in VxWorks drei Wind Semaphores

binary Schnellster Typ, allgemein einsetzbar, optimiert für Tasksynchronisation, kann auch für wechselseitigen Ausschluss eingesetzt werden.

mutual exclusion Spezieller binary-Typ, optimiert für inhärente Probleme wechselseitigen Ausschlusses: priority inheritance, sicheres Löschen, Rekursion.

counting Zähl-Semaphore, optimiert für Zugriff auf mehrere Instanzen einer Ressource.

Semaphor-Kontrolle

Semaphorerzeugung

```
// Allokation und Initialisierung eines
// binary Semaphor
semBCreate(int options, SEM_B_STATE initialState );
// Allokation und Initialisierung eines Semaphor
// fuer wechselseitigen Ausschluss
semMCreate(int options);
// Allokation und Initialisierung eines
// Zaehl-Semaphor
semCCreate(int options, int initialCount);
```

Semaphor-Kontrolle (Forts.)

Semaphorterminierung und -setzen

```
// Terminierung des Semaphor
semDelete(SEM_ID semId);
// "Nehmen" eines Semaphor
semTake(SEM_ID semId, int timeout);
// "Geben" eines Semaphor
semGive(SEM_ID semId);
// Deblockieren aller Tasks,
// die auf ein Semaphor warten
semFlush(SEM_ID semId);
```

Beispiel Binary Semaphor

Semaphor “nehmen”

- Ein binary Semaphor ist entweder verfügbar oder nicht verfügbar.
- Falls ein Task ein Semaphor mit semTake() “nimmt” und das Semaphor verfügbar ist, wird das Semaphor nicht verfügbar und der Task führt seine Ausführung normal fort.
- Ist das Semaphor nicht verfügbar, wird der Task blockiert und wartet in einer Warteschlange darauf, dass das Semaphor verfügbar wird.

Beispiel Binary Semaphor (Forts.)

Semaphor “geben”

- Falls ein Task ein Semaphor mit semGive() “gibt” und das Semaphor bereits verfügbar ist, hat die Operation keinen Effekt.
- Falls das Semaphor nicht verfügbar ist und kein Task auf dieses Semaphor wartet, wird es verfügbar.
- Ist das Semaphor nicht verfügbar und mindestens ein Task wartet darauf, wird der erste Task in der entsprechenden Warteschlange deblockiert und das Semaphor bleibt nicht verfügbar.

Experiment

Im Beispielprogramm VxWorks_semaphore.c konkurrieren zwei Tasks taskOne und taskTwo um das Setzen einer Variable global. Das Programm soll den Wert von global zwischen 0 und 1 hin- und herschalten. TaskOne setzt sie auf 1, TaskTwo auf 0.

Aufgabe: bringen Sie das Beispielprogramm VxWorks_signal.c zum laufen.

Aufgabe: Entfernen Sie im Beispielprogramm die mit Note 1 und Note 2 verbundenen Programmzeilen, kompilieren und führen Sie das geänderte Programm aus. Erklären Sie die Unterschiede im Programmverhalten.

Aufgabe: Schreiben Sie ein Programm, das das Hin- und Herschalten der Variable global mit einem Zähl-Semaphor realisiert.

VxWorks_semaphore.c

```c
/* includes */
#include "vxWorks.h"
#include "taskLib.h"
#include "semLib.h"
#include "stdio.h"

/* function prototypes */
void taskOne(void);
void taskTwo(void);

/* globals */
#define ITER 10
SEM_ID semBinary;
int global = 0;
```

VxWorks_semaphore.c (Forts)

```
void binary(void){
int taskIdOne, taskIdTwo;

/* create semaphore with semaphore available
 and queue tasks on FIFO basis */
semBinary = semBCreate(SEM_Q_FIFO, SEM_FULL);

/* Note 1: lock the semaphore for scheduling purposes */
semTake(semBinary,WAIT_FOREVER);

/* spawn the two tasks */
taskIdOne = taskSpawn("t1",90,0x100,2000,
      (FUNCPTR)taskOne,0,0,0,0,0,0,0,0,0,0);
taskIdTwo = taskSpawn("t2",90,0x100,2000,
      (FUNCPTR)taskTwo,0,0,0,0,0,0,0,0,0,0);
}
```

VxWorks_semaphore.c (Forts)

```
void taskOne(void){
int i;
for (i=0; i < ITER; i++)
  {
  semTake(semBinary,WAIT_FOREVER); /* wait indefinitely
                                      for semaphore */
  printf("I␣am␣taskOne␣and␣global␣=␣%d.....................\n",
         ++global);
  semGive(semBinary); /* give up semaphore */
  }
}
```

VxWorks_semaphore.c

```
void taskTwo(void){
int i;
semGive(semBinary); /* Note 2: give up semaphore
                       (a scheduling fix) */
for (i=0; i < ITER; i++)
  {
  semTake(semBinary,WAIT_FOREVER); /* wait indefinitely
                                      for semaphore */
  printf("I␣am␣taskTwo␣and␣global␣=␣%d----------------------\n",
         --global);
  semGive(semBinary); /* give up semaphore */
  }
}
```