REAL TIME OPERATING SYSTEM PROGRAMMING-I: μC/OS-II and VxWorks

Lesson-11:

VxWorks Signal and Semaphore Functions

1. Signal Function

Signal (an Exception or Interrupt) handling Functions

- sigNum

– identifies a signal

- void sigHandler (int sigNum);

–to declares a signal servicing routine for a signal identified by *sigNum*

- signal (sigNum, sigISR)

–to registers a signal sigNum with a signal servicing routine sigISR

- Refer Section 9.3.3.4

2. Semaphore Functions

Semaphore Creation Functions

- semBCreate ( ) Creates a binary semaphore. Example 9.21 Step 4
- semMcreate ( ) Creates a mutex semaphore. Example 9.22 Step 2
- semC-Create Creates a counting semaphore. Example 9.23 Step 5

Semaphore Deletion and Flush Functions

- semDelete ( )

–to deletes a semaphore

- semFlush ( )

–to resume all waiting blocked tasks.

[Semaphore value for eah waiting task becomes 1]

Semaphore Take and Give Functions

semTake ( ) Takes a semaphore. Example 9.22 Step 6

semGive ( ) Releases a semaphore.[Example 9.21 Step 7, 9.22 Step 7 and 9.23 Step 12]

OPTIONS in Semaphore Functions

(1)SEM_Q_FIFO

for specifying that take the semaphore in FIFO mode (task blocked first get that first)

(2) SEM_Q_PRIORITY

for specifying that take the semaphore in priority mode (higher priority blocked task get that first)

Semaphore Create for Event Flag

- SEM_ID semBCreate (*options*, *initialState*)'

  –to create an ECB for binary semaphore *SEM_ID* with options and initialState value

- Refer section 9.3.4.1 and example 9.21 Step 4

Mutex Creation Statements

```
SEM_ID semMKeyID;
semMKeyID = semBCreate
(SEM_Q_PRIORITY, SEM_FULL);
/*creates a Mutex using a binary sem.*/
```

- Refer section 9.3.4.5 and example 9.22

Step 2

Example

```
SEM_ID semMReadPortAKey;
semMReadPortAKey = semMCreate
(SEM_Q_PRIORITY |
SEM_INVERSION_SAFE |
SEM_DELETE_SAFE);
/* To initial count = SEM_FULL, which means 1 and
create a mutex semaphore with priority option for
taking that, priority inversion safe, and deletion
safe */
```

Example

- SEM_ID semCCreate (*options*, unsigned byte *initialCount*)

–to create an ECB pointed by the SEM_ID.

- SEM_ID semCID;

•. SEM_ID = semCCreate (SEM_Q_PRIORITY, 0);

/* To initial count = 0 and create a counting semaphore with priority option for taking that*/

POSIX Semaphores

- semPxLibInit ( );/* initializes the VxWorks library to permit use of these Functions */
- sem_open ( );/* initialize a semaphore*/
- sem_close ( ); /*close the semaphore*/
- sem_unlink ( ); /*remove a semaphore*/

POSIX Semaphore Functions

- sem_post ( );
- sem_wait ( );
- sem_unlock;
- sem_lock a semaphore;
- sem_trywait ( ); /* to lock a semaphore if not locked*/
- sem_getvalue ( ) /* to retrieve the value of semaphore /*

Init and Destroy

- sem_init ( ) and sem_destroy ( ) initialise and destroy an unnamed semaphore.

Destroy means de-allocate associated memory with its ECB. This effect is the same as first closing a semaphore and then unlinking it by sem_close and sem_unlink functions, respectively.

VxWorks Semaphore Special Features

(i) Options of protection from priority inversion and task deletion.

(ii) Single task may take a semaphore multiple times and recursively.

VxWorks Semaphore Special Features

(iii) Mutually exclusive ownership can be defined.

(iv) Two options, FIFO and task priority for semaphores wait by multiple tasks.

Summary

We learnt

- signal-servicing routines
- A signal-servicing routine is a C function and that executes on signal (interrupt or exception) in a task.
- A connect function connects the signal number with the signal ISR

We learnt

- two ways in which a pending task among the pending tasks can unblock. One is as per task priority and another is as a FIFO when accepting (taking) an IPC

We learnt

- three different semaphore functions for use as IPC for event flag, for resource key and counting semaphore.
- supports POSIX semaphores.

End of Lesson-11 on VxWorks Signal and Semaphore functions