Inter-Process Communication and Synchronization of Processes, Threads and Tasks:

Lesson-6: Concept of Semaphore as an event signaling variable or notifying variable

Semaphore as an event signaling variable or notifying variable

Semaphore as an event signaling variable or notifying variable

- Suppose that there are two trains.
- Assume that they use an identical track.
- When the first train A is to start on the track, a signal or token for A is set (true, taken) and
- same signal or token for other train, B is reset (false, not released).

OS Functions for Semaphore as an event signaling variable or notifying variable

- OS Functions provide for the use of a semaphore for signaling or notifying of certain action or notifying the acceptance of the notice or signal.
- Let a binary Boolean variable, *s*, represents the semaphore.

OS Functions for Semaphore as event notifying variable

- The taken and post operations on *s*— *(i)* signals or notifies operations for communicating the occurrence of an event and (ii) for communicating taking note of the event.
- Notifying variable *s* is like a token — (i) acceptance of the token is taking note of that event (ii) Release of a token is the occurrence of an event

Binary Semaphore

Binary Semaphore

- Let the token (flag for event occurrence) *s* initial value = 0
- Assume that the *s* increments from 0 to 1 for signaling or notifying occurrence of an event from a section of codes in a task or thread.
- When the event is taken note by section in another task waiting for that event, the *s* decrements from 1 to 0 and the waiting task codes start another action.

Binary Semaphore…

- When s = 1— assumed that it has been released (or sent or posted) and no task code section has taken it yet
- When s = 0 — assumed that it has been taken (or accepted) and other task code section has not taken it yet

Binary Semaphore use in ISR and Task

- An ISR can release a token.
- A task can release the token as well accept the token or wait for taking the token

Example of ACVM

Uses in ACVM

- *Chocolate delivery task* — after the task delivers the chocolate, it has to notify to the display task to run a waiting section of the code to display "*Collect the nice chocolate. Thank you, visit again*".
- The waiting section for the displaying the thank you message takes this notice and then it starts the display of thank you message

Uses in ACVM…

- Assume OSSemPost ( )─ is an OS IPC function for posting a semaphore
- OSSemPend ( ) ─ another OS IPC function for waiting for the semaphore.

Uses in ACVM…

- Let *sdispT* is the binary semaphore posted from *Chocolate delivery task* and taken by a *Display task* section for displaying thank you message.
- Let *sdispT* initial value = 0.

Uses in ACVM…

static void Task_Deliver *(void *taskPointer)* **{**

.

while (1) {

.

/* Codes for delivering a chocolate into a bowl. */

.

OSSemPost (*sdispT*) /* Post the semaphore *sdispT*. This means that OS function increments *sdispT* in corresponding event control block. *sdispT* becomes 1 now. */

.

};

Uses in ACVM…

static void Task_Display *(void *taskPointer)* **{**

.

while (1) {

.

OSSemPend (*sdispT*) /* Wait *sdispT*. means wait till *sdispT* is posted and becomes 1. When *sdispT* becomes 1 and the OS function decrements *sdispT* in corresponding event control block. *sdispT* 0 now. Task then runs further the following code*/

/* Code for display "*Collect the nice chocolate. Thank you, visit again"* */

.

};

Summary

We learnt

- Semaphore provides a mechanism to let a section of the task code wait till another finishes an action (finish running of the codes).

We learnt

- Provides a way of signaling an event occurrence.
- Provides a way of signaling taking of a note of the event. Semaphore increments when posted (sent or released) and decrements when accepted or taken by waiting task section.

We learnt

- A waiting task-section is notified to start on sending (posting or releasing) the semaphore.
- The section starts on taking the semaphore

End of Lesson 6 of Chapter 7