REAL TIME OPERATING SYSTEMS

Lesson-3:

Timer Functions

1. Timer Functions

Timer Functions

- Real time clock– system clock, on each tick SysClkIntr interrupts
- Based on each SysClkIntr interrupts– there are number of OS timer functions

Timer functions at OS

- OS_TICK_PER_SEC– to set the system clock ticks and hence **SysClkIntr interrupts per s**
- OSTickInit ( )– to initiate system clock ticks

Timer functions at OS

- OSTimeDelay ( )─ to delay the process making call by a fixed number of system clock ticks specified in argument
- OSTimeDelay-Resume ( )─ to resume a delayed process specified in the argument

Timer functions at OS

- OSTimeSet ( )─ to set the *counts* of system clock ticks
- OSTimeGet ( ) ─ to read the counts of system clock ticks to find the time interval from the previous read or write of the *counts*

Timer functions at OS

- OSSemPend (semVal, twait, *semErr)─ to wait for a semaphore release
- OSMboxPend (semVal, twait, *mboxErr) ─ to wait for a message in mailbox (wait for message pointed not Null)

2. System clock ticks and hence SysClkIntr interrupts

SysClkIntr interrupts

- Before servicing of SysClkIntr, the presently running task or thread or process context saves on the TCB or thread stack or PCB data structure,
- System switches to supervisory mode
- SysClkIntr service routine call the OS to increment system time and to find the new messages or IPCs, which the OS event control blocks received from the system call for the IPC functions,

SysClkIntr service routine

- call the OS functions (i) to increment system time and (ii) to find the new messages or IPCs, which the OS event control blocks received from the system call for the IPC functions,
- (iii) call the OS function as per the message or system call

SysClkIntr service routine…

- (iv) Then OS either selects the same task or selects new task or thread [by preemption in case of preemptive scheduling and switches the context to the new one, and
- Then after return from the interrupt the new task runs from the code, which was blocked from running

2. Examples of Timer functions

OS_TICK_PER_SEC

- # define OS_TICK_PER_SEC 100
  /*μCOS-II function to define the number of ticks per second = 100 before the beginning of the main ( ) and the initiating the OS by OSInit ( ) function*

OSTickInit ( )

- OSTickInit ( ) /*µCOS-II function to initiate the defined number of ticks per second after the beginning of the first task and creating all the tasks to which the context will be switched by the OS on the tick. It initiates SysClkIntr interrupts every 10 ms when number of ticks = 100 per s*/.

OSTimeDelay (*n*)

- OSTimeDelay (*n*) by period equal to period of *n* clock ticks
- OSTimeDelay (100);

/* Code for finding the coins amount after every 100 clock ticks, which means every 1 s*/

; }

OSTimeSet (Count) and OSTimeGet ()

- count =0; OSTimeSet (Count); while (count < = 6000) /* While loop waits for the coins amount upto 60000 ms = 1 minute */

{Count = OSTimeGet ( );

- /*Calling OSTimeGet ( ), finding the count, *count1*, running a section of codes and then calling OSTimeGet ( ), finding the new count, *count2* give us the interval, T spent by the system in between the two function calls of OSTimeGet ( ). T = (count2 – count1) × interval between two clock ticks gives the interval*/

Summary

We learnt

- SysClkIntr interrupts
- OS_TICK_PER_SEC
- OSTickInit ( )
- OSTimeDelay ( )
- OSTimeDelay-Resume ( )

We learnt

- OSTimeSet ( )
- OSTimeGet ( )
- OSSemPend (semVal, twait, *semErr)
- OSMboxPend (semVal, twait, *mboxErr)

End of Lesson 3 of Chapter 10