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

Lesson-9:

Basic RTOS Functions in VxWorks

RTOS Programming Tools: MicroC/OS-II and VxWorks

Lesson 9:

Basic RTOS Functions in VxWorks

1. VxWorks Features

WindRiver® VxWorks

- High-performance, Unix-like, multitasking Environment scalable and hierarchical RTOS
- Host and target based development approach
- Supports ‘Device Software Optimization ― a new methodology that enables development and running of device software faster, better and more reliably

VxWorks RTOS Kernel…

- VxWorks 6.x processor abstraction layer
- The layer enables application design for new versions later by just changing the layer-hardware interface
- Supports advanced processor architectures– ARM, ColdFire, MIPS, Intel, SuperH,…

VxWorks RTOS Kernel…

- Hard real time applications
- Supports kernel mode execution of tasks
- Supports open source Linux and TIPC (transparent inter process communication) protocol

VxWorks RTOS Kernel…

- Provides for the preemption points at kernel
- Provides preemptive as well as round robin scheduling,
- Support POSIX standard asynchronous IOs
- Support UNIX standard buffered I/Os

VxWorks RTOS Kernel…

- PTTS 1.1 (Since Dec. 2007)
- IPCs in TIPC for network and clustered system environment
- POSIX 1003.1b standard IPCs and interfaces additional availability
- Separate context for tasks and ISRs [Each task has a separate TCB, while ISRs a common stack]

VxWorks RTOS Kernel…

- Schedules the ISRs separately and has special functions for interrupt handling
- Watchdog timers
- Virtual I/O devices including the pipes and sockets (Sections 7.14 and 7.15)
- Virtual Memory Management functions

VxWorks RTOS Kernel…

- Power management functions that enhance the ability to control power consumption
- Automatic detection and reporting of common memory and other errors
- Interconnect functions that support large number of protocols
- APIs for IPv4/IPv6 dual mode stack

Host-Target Development Approach

- Host Windows, Linux or Unix for Embedded Development and cross compiled for target system processor
- RTOS ROM resident code downloaded to the using TCP/IP or serial port to a target board
- Target has no virtual memory support and needed kernel functions are at the target ROM

Scalability

- Scalable OS – only needed OS functions become part of the application codes
- Configuration file includes the user definitions for the needed IPC functions needed

Hierarchical

- RTOS kernel extendibility and interfaces hierarchy includes timers, signals, TCP/IP Sockets, queuing functions library, NFS, RPCs, Berkeley Port and Sockets, Pipes, Unix compatible loader, language interpreter, shell, debugging tools and linking loader for Unix.

Protected Environment

- Protection features – for example, if a task is expecting a message from another task, which is being deleted by using the task-delete function, then RTOS inhibits the deletion
- No priority inversion problem—

the task gets an inherited priority when option of priority inheritance selected

Header Files

- VxWorks.h – header file
- kernelLib.h – kernel library functions header file
- taskLib.h – tasks library functions header file
- sysLib.h system library functions header file

VxWorks Basic Functions

- System Level – OS initiate, start, system timer clock rate set, ISR enter and exit, enable and disable
- Task Service Functions – initiate, resume, activate, run, suspend, (now or after delay)
- Task control functions

VxWorks Basic Functions…

- IPCs – Semaphore, Queue and Pipes, POSIX IPCs
- No Mailbox
- Queue permit array of messages
- Network Functions
- IO Functions

2. Signal (Software interrupt from task) Handling

Signal

- IPC signal used for exception handling or handling software interrupt event
- Signal-servicing routine─ a C function, which executes on occurrence of an interrupt or exception.
- Signal connect function connects the function with an interrupt vector

3. Semaphore Functions

Semaphore functions for Synchronization

- event signal flag,
- mutually exclusive access using resource key (mutex) and
- counting mechanism using three type of semaphores in the tasks and ISRs
- P-V semaphore functions when POSIX library included

Two ways in which a pending task among the pending tasks unblock

- Provides for– (a) as per task priority (b) as a FIFO, when accepting or taking an IPC

4. Queues Functions

Queue…

- Instead of queuing the message pointers in µC/OS-II , provides for queuing of the messages.
- Queues can be used for priority posting of message using *post front* as in µC/OS-II

Queue

- Provides for two ways in which a pending task among the pending tasks can unblock – (a) as per task priority (b) as a FIFO, when accepting or taking an IPC

5. Virtual device Functions

5. Virtual device Functions

- Pipe Drivers for inter-process communications as an I/O virtual device
- Network-transparent sockets.
- Network drivers for shared memory and Ethernet.
- RAM "disk" drivers for memory resident files
- files

6. Task Service Functions

Task functions

- Task creation and activation distinct states
- Functions for the task creating, running, waiting, suspending (inhibiting task-execution) and resuming, spawning (creating followed by activating), task-pending cum suspending and pending cum suspension with timeout functions

7. VxWorks Functions naming Basics

VxWorks Naming Basics

- No OS or OS_ prefix for functions
- For example, taskInit ( ) - a VxWorks function, which initiates a task
- Prefix VX_ for the options and macros
- For example, VX_PIVATE_ENV VX_NO_STACK_FILL|

Summary

We learnt

- VxWorks basic features in the functions
- high performance,
- scalable and hierarchical
- OS initiate and start
- scheduling
- error handling
- system clock and service
- time delay

We learnt

- task,
- memory
- pipes,
- files
- network
- IPC

End of Lesson 9 on

Basic RTOS Functions in VxWorks