e-PG Pathshala

Subject : Computer Science
Paper: Embedded System
Module: RTOS – Basic concept
Module No: CS/ES/24
Quadrant 1 – e-text

In this lecture an introduction to real time operating system (RTOS) will be given. The types and functions of RTOS will be discussed elaborately. The different RTOS software will also be explored.

1. Real time in operating systems

The operating system provides valuable services that the activities (processes) uses to efficiently and effectively interact. In particular, these services enable us to communicate with the proper counterparts. Importantly, these services allows the activities to use the available resources efficiently (e.g. signal & wait avoids the need for polling). However when we deal with real time applications, the resource management has to be done with additional constraints. Hence we need a class of operating systems called 'Real Time Operating Systems'.

2. RTOS (Real Time Operating System):

A real time operating system (RTOS) is a multitasking operating system for the applications with hard or soft real time constraints. Real-time constraint means the constraint on occurrence of an event, system expected response and latency to the event. It also provides perfection, correctness, protection and security features of any kernel of OS when performing multiple tasks. RTOS responds to inputs immediately i.e in real time. The task is completed within a specified time delay. For example, in case of traffic control signal or a nuclear reactor or an aircraft, the RTOS has to respond quickly.

The main reasons for going to RTOS are effective use of drivers available with RTOS and we can focus on developing the application code rather than creating and maintaining a scheduling system. RTOS also supports multi-threading with the synchronization mechanism. The developed application code can be portable to other CPUs also. Resource allocation and management processes are also handled properly. We can add new features without affecting the high priority functions or tasks. RTOS also supports the upper layer protocols such as TCP/IP, USB, Flash Systems, Web Servers, CAN protocols, Embedded GUI, SSL, SNMP.

2.1 Requirements for RTOS

The following are the requirements for a good RTOS.

The timing behavior of the OS must be predictable because of effective time allocation and deallocation before starting of any process or task execution. For all the services of the OS, there is an upper bound on the execution time. Effective scheduling policies are used for running of multiple tasks and this scheduling policy is to be a predefined one. The period during which the interrupts are disabled must be short to avoid delay (Time slicing mechanism is used to reduce the time delay). OS should be aware of task deadlines prior to the execution of task. Frequently, the OS should provide precise time services with high resolution. Then the OS must be fast to execute the allocated task without delay.

2.2 Types of RTOS

The RTOS are of two types namely soft real time RTOS and hard real time RTOS. The soft real-time tasks are performed as fast as possible. In soft real time task, late completion of jobs is undesirable but not fatal. The performance of the system degrades as more and more jobs miss deadlines. An example for soft real time task is an Online Database.

Hard real-time tasks have to be performed on time, and failure to meet deadlines is fatal. An example for hard real time task is a flight control system.In the hard real time system meeting the deadline is very important, if deadline is not met the system performance will fail.

2.3 Architecture of RTOS:

The basic architecture of a multitasking RTOS consists of a) Program interface b) The Kernel c) Device Drivers and d) Optional service modules. Figure 1 shows the architecture of RTOS.

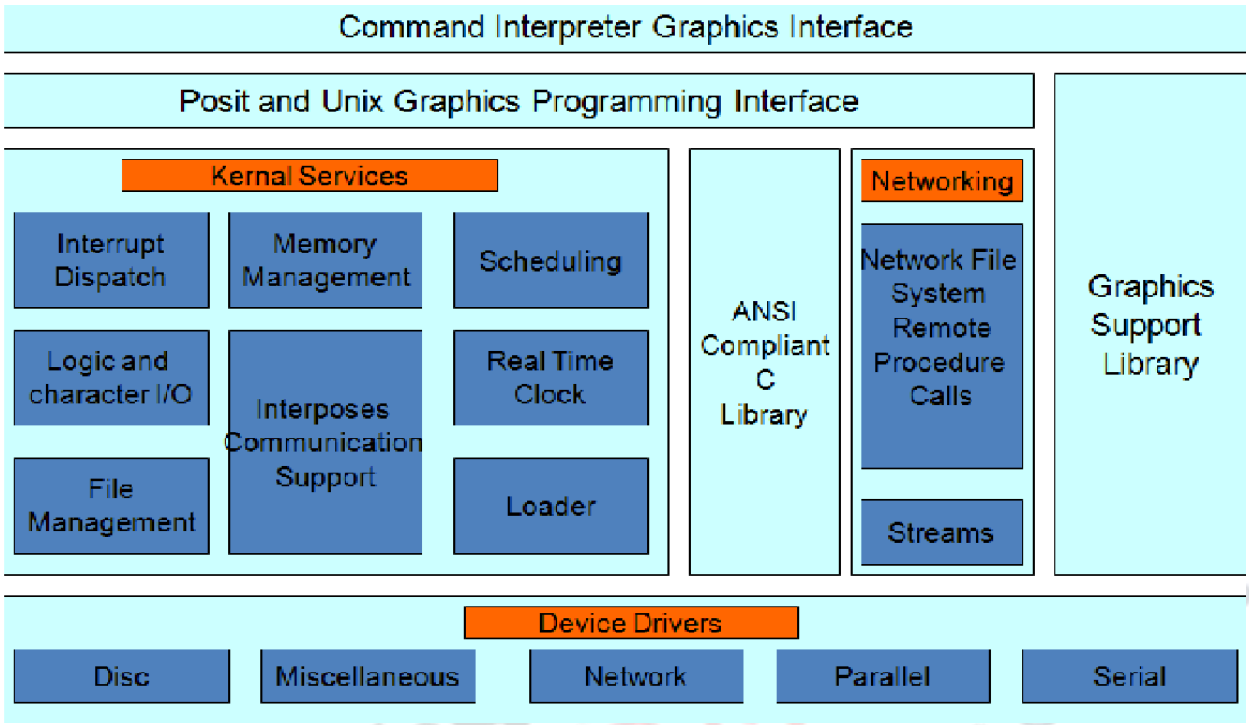

Figure 1: Architecture of RTOS

Kernel Services

The "kernel" of a real-time operating system ("RTOS") provides an "abstraction layer" that hides the hardware details of the processor from the application software. The Kernel provides an interrupt handler, task scheduler, resource sharing flags and memory management. The primary function of the kernel is to process interrupts that external and internal events cause. For embedded applications Kernel size should be small and it should fit into the ROM. Hence sophisticated features can be removed. The scheduler mainly sets up the order of execution of application code. Some of the kernel services are explained below.

a) Memory Management

The processes carried out in memory management are Memory allocation, Deallocation and Management. It also restricts the memory access region for a task. There may be dynamic memory allocations also.

b) File Management

A file is a named entity on a magnetic disc, optical disc or system memory. A file contains the data, characters and text. It may also have mix of these. Each OS has different abstractions of a file. The File manager appropriately keeps track of the files.

c) Logical and Character I/O

A logical I/O, also known as a buffer I/O, refers to reads and writes of buffers in the buffer cache. When a requested buffer is not found in memory, the I/O system performs a physical I/O to copy the buffer from either the flash cache or disk into memory, and then a logical I/O to read the cached buffer. Character I/O consists of a sequence of bytes transferred serially.

d) Real Time Clock

A real-time clock (RTC) is a computer clock which keeps track of the current time.

A good RTOS should have the following characteristics:

It must support multitasking capabilities. A real time application is divided into multiple tasks. The separation of tasks helps to keep the CPU busy. It should have Short Interrupt Latency. The interrupt latency is equal to the sum of time taken to get the interrupt signal to the processor, the time taken to complete the current instruction and the time for executing the system code in preparation for transferring the execution to the device's interrupt handler.

The RTOS must provide Fast Context Switch. The context switch time is the time between the OS recognizing that the awaited event has arrived and the beginning of the waiting task (dispatch latency). This switching time should be minimum. It must manage memory properly i.e it must have control on memory management.

The OS should provide way for task to lock its code and data into real memory so that it can guarantee predictable response to an interrupt. It must do a proper scheduling. The OS must provide facility to schedule properly time constrained tasks. It must support the Fine granularity Timer Services. Millisecond resolution is bare minimum. Microsecond resolution is required in some cases. It should perform Inter Task Communication Mechanism. The inter task communication is performed by using Message queues, shared memory, semaphores, event flags.The RTOS must have the characters like consistent, reliable, scalable, predictable and better performance.

3. Functions of RTOS

The important functions done by RTOS are task management, scheduling, resource allocation and interrupt handling.

3.1 Task management:

In Real Time Applications the Process is called as Task which takes execution time and occupies memory. The task management is the process of managing tasks through its life cycle. Task will have different states. The states of task are Pended, Ready, Delayed, Suspended, and Run. Figure 2 shows the state diagram of task.

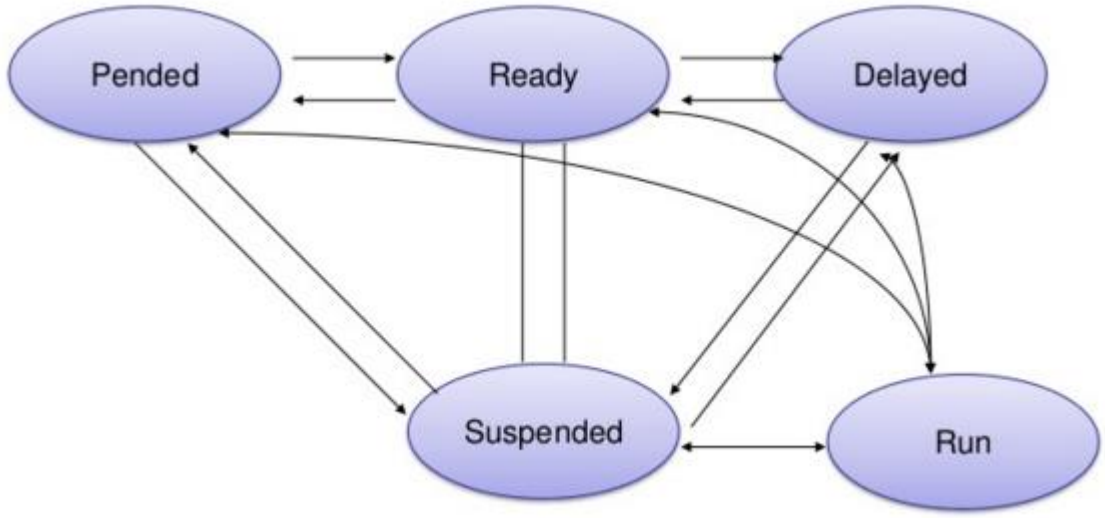

Figure 2: State diagram of task

3.1.1 Task/Process States:

Each Task/ process will be in any one of the states. The states are pended, ready, suspended, delayed and run. The scheduler will operate the process which is in ready state. At any time only one process is in run state. Transitions to and from the Ready queue are affected as a part of the execution of the RTOS services or as a result of timing events.

3.1.2 Typical Task Operations:

The important task operations are creating and deleting tasks, controlling, task scheduling and obtaining task information.

3.2. Scheduling in RTOS:

In order to schedule task, information about the task must be known. The information of task are the number of tasks, resource requirements, execution time and deadlines. Based on the system, if it is deterministic, better scheduling algorithms can be devised.

3.2.1 Scheduling Algorithms in RTOS:

A scheduler may aim at one of many goals, for example, maximizing throughput , minimizing response time, or minimizing latency and maximizing fairness. In practice, these goals often conflict (e.g. throughput versus latency), thus a scheduler will implement a suitable compromise. Preference is given to any one of the concerns mentioned above, depending upon the user's needs and objectives.

The scheduling algorithms are broadly classified into priority based and non priority based scheduling. The Priority based scheduling is based on Greedy / List / Event Driven.The non priority based are based on first come first served.

3.2.2 Non-Priority based Scheduling algorithms

Non-Priority based algorithms will give importance like first come first served basis. The simplest best-effort scheduling algorithms are clock driven and weighted round-robin. In clock driven scheduling, the information about all parameters of the jobs must be known in advance. The parameters of the jobs are release time, execution time and deadline. The schedule can be done offline or at some regular time instances. The run time overhead will be minimal. It may not be suitable for all applications.

In round robin, jobs are scheduled in FIFO manner. Time slot is given to jobs and it is proportional to its weight. Each job will execute in its time slot. If it is not completed within time slot, the job is removed and it is kept in the queue. When it gets its turn, the remaining portion of the job is to be executed.

3.2.3 Priority based Scheduling (Greedy/List/Event Driven):

In this scheduling, processor is never left idle when there are ready tasks. The processor is allocated to processes based on priorities. Priorities are of two types - one is static and other is dynamic. In static priorities are assigned at designed time. Once the priorities are assigned during run time changing is not possible. The dynamic priority scheme will allow change of priority at run time also.

In Earliest deadline first (EDF) scheme, the processor with earliest deadline has been assigned highest priority. In Least slack time first (LSF) scheme the priority is assigned based on slack time, where slack time is difference between relative dead line and execution left (these will be explained in detail in module-26).

This scheduling scheme is mainly used for periodic tasks. Task priority is proportional to the inverse of the period of the task. All these scheduling algorithms will be discussed in detail under the heading scheduling of RTOS (module-26).

3.3. Resource Allocation in RTOS:

Resource allocation is necessary for any application to be run on the system. when an application is running, it requires the OS to allocate certain resources for it to be able to run. Such resources could have access to a section of the computer's memory, data in a device interface buffer, one or more files, or the required amount of processing power. In RTOS, the scheduling algorithms discussed above are used for the resource allocation. The resources are allocated based on the weighted round robin and priority based scheduling policies.

4. Interrupts handling in RTOS:

An interrupt is a signal from a device attached to a computer or from a program within a computer. It stops the main program and responds for the event which is interrupted. Interrupts cause the processor to suspend the operations which it is doing and execute the code(Interrupt Service Routine) that will respond to the event that caused the interrupt. The time taken to handle the interrupt, that is, interrupt latency should be very small. Interrupts should be disabled for minimum possible time. Interrupt Service Routines(ISR) should have higher priorities over the RTOS functions and the tasks. An ISR should not wait for a semaphore, mailbox message or queue message. An ISR should not also wait for mutex. It has to wait for other critical section code to finish before the critical codes in the ISR can run.

5. Applications of RTOS:

- Almost all the modern telecommunication systems make use of RTOS.
- Radar systems, network switching control systems, satellite monitoring systems, satellite launch-control and maneuvering mechanisms, global positioning systems all have their roots in RTOS.
- Now a day's RTOS are increasingly finding use in strategic and military operations. These are used in guided missile launching units, track-and-trace spy satellites, etc.

6. Comparison of RTOS:

There are many RTOS programming tools available commercially. Some of them used are Vxworks, pSOS and eCOS. The comparison of the above mentioned tools are given below in figure 3.

	VXWorks	pSOS	eCos
Scheduler	Preemptive	Preemptive	Preemptive
Synchronization mechanism	No condition variable	Y	Y
POSIX support	Y	Y	Linux
Scalable	Y	Y	Y
Custom hw support	BSP	BSP	HAL, I/O package
Kernel size	-	16KB	-
Multiprocessor support	VxMP/ VxFusion (accessories)	PSOS+m kernel	Y/only basic support (SMP)

Figure 3: Comparison of RTOS

Let us discuss one of the RTOS programming tool in detail(VxWorks). VxWorks is the most established and most widely deployed device software operating system. Figure 3 shows the comparison of RTOS. Currently there are more than 300 million devices that are VxWorks enabled. The core attributes of VxWorks, includes high performance, reliability, determinism, low latency and scalability. VxWORKS is Created by Wind River, it come up with different versions, the current version is VxWorks 6.0. It has enhanced error management. It is having backward compatibility to previous versions features for exception handling and template support.
Scheduling - It uses preemptive priority with round robin scheduling to accommodate for both Real time processes and Non-real time processes.Memory Protection - It uses MMU based memory protection. Reduced Context Switch time -In order to reduce context switch it follows the following principles. It saves only those register windows that are actually in use. When a task's context is restored, only the relevant register window is restored. To increase response time, it saves the register windows in a register cache.

7. Summary

In this lecture we discussed about the basics of RTOS. Types and functions of RTOS are also discussed elaborately. Different RTOS software in embedded fields are indicated and given an introduction to Vxworks software.

8. References:

1. Steve Heath "Embedded systems design" second edition,2003 Newnes (ELSEVIER).
2. Wayne Wolf, "Computers as components: Principles of embedded computing systems design"; Second edition, published by Morgan Kaufmann series(2008).

3. http://www.slideshare.net/Tech_MX/real-time-os.
4. www.pantechsolutions.net.
5. http://www.slideshare.net/pantechsolutions.