UNIT - V

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

Real Time Operating Systems

- Introduction, Task and Task States, Tasks and Data Reentrancy
- Semaphores and Shared Data, Basic Design Principles
- Interprocess Communication, Message Queues, Mail boxes & Pipes
- Timer Functions and Events
- Memory Management and Interrupt Routines in an RTOS Environment

RTOS Basics

Operating system (OS) :

An Operating system (OS) is a piece of software that controls the overall operation of the System. It acts as an interface between hardware and application programs.

It facilitates the user to format disks, create, print, copy, delete and display files, read data from files, write data to files, control the I/O operations, allocate memory locations and process the interrupts etc.

In a multiuser system it allows several users to share the CPU time, share the other system resources and also avoids the interference of different users in sharing the resources etc. Hence the OS is also known as a resource manager.

An Operating system (OS) is nothing but a collection of system calls or functions. An OS typically provides multitasking, synchronization, Interrupt and Event Handling, Input/Output, Inter-task Communication, Timers and Clocks and Memory Management.

Popular Operating Systems:

MS-DOS, Windows (Microsoft), Linux(Open source), Unix (Multi user-Bell Labs), MacOS, Android (Mobile).

RTOS Basics .....

REAL TIME SYSTEMS:

Real-time systems are those systems in which the correctness of the system depends not only on the Output, but also on the time at which the results are produced (Time constraints must be strictly followed).

Real time systems are two types:

(i) **Soft real time systems** (ii) **Hard real time systems**

- Soft Real Time system is one in which the performance of the system is only degraded but, not destroyed if the timing deadlines are not met.
  - Ex: Air conditioner, TV remote or music player, Bus reservation, automated teller machine in a bank, Lift, etc.
- Hard Real Time system is one in which the failure to meet the time dead lines may lead to a complete catastrophe or damage to the system.
  - Ex: Air navigation system, Nuclear power plant, Failure of car brakes, Gas leakage system, RADAR operation, etc.

RTOS Basics .....

REAL TIME OPERATING SYSTEM (RTOS):

It is an operating system that supports real-time applications by providing logically correct result within the deadline set by the user. A real time operating system makes the embedded system into a real time embedded system.

A Real-Time Operating System (RTOS) comprises of two components, viz., "Real-Time" and "Operating System".

Basic Structure is similar to regular OS but, in addition, it provides mechanisms to allow real time scheduling of tasks

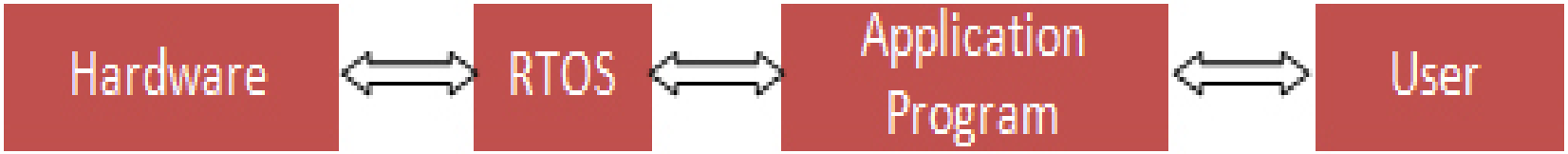


Fig. Real time embedded system with RTOS

- All the embedded systems are not designed with RTOS. Low end application systems do not require the RTOS but only High end application oriented embedded systems which require scheduling alone need the RTOS.
- For example an embedded system which measures Temperature or Humidity etc. do not require any operating system.
- Where as a Mobile phone, RADAR or Satellite system used for high end applications require an operating system.

RTOS Basics .....

RTOS Classification:

RTOS specifies a known maximum time for each of the operations that it performs. Based upon the degree of tolerance in meeting deadlines, RTOS are classified into following categories

- Hard real-time: Degree of tolerance for missed deadlines is negligible. A missed deadline can result in catastrophic failure of the system
- Firm real-time: Missing a deadly ne might result in an unacceptable quality reduction but may not lead to failure of the complete system
- Soft real-time: Deadlines may be missed occasionally, but system doesn't fail and also, system quality is acceptable

RTOS Features:

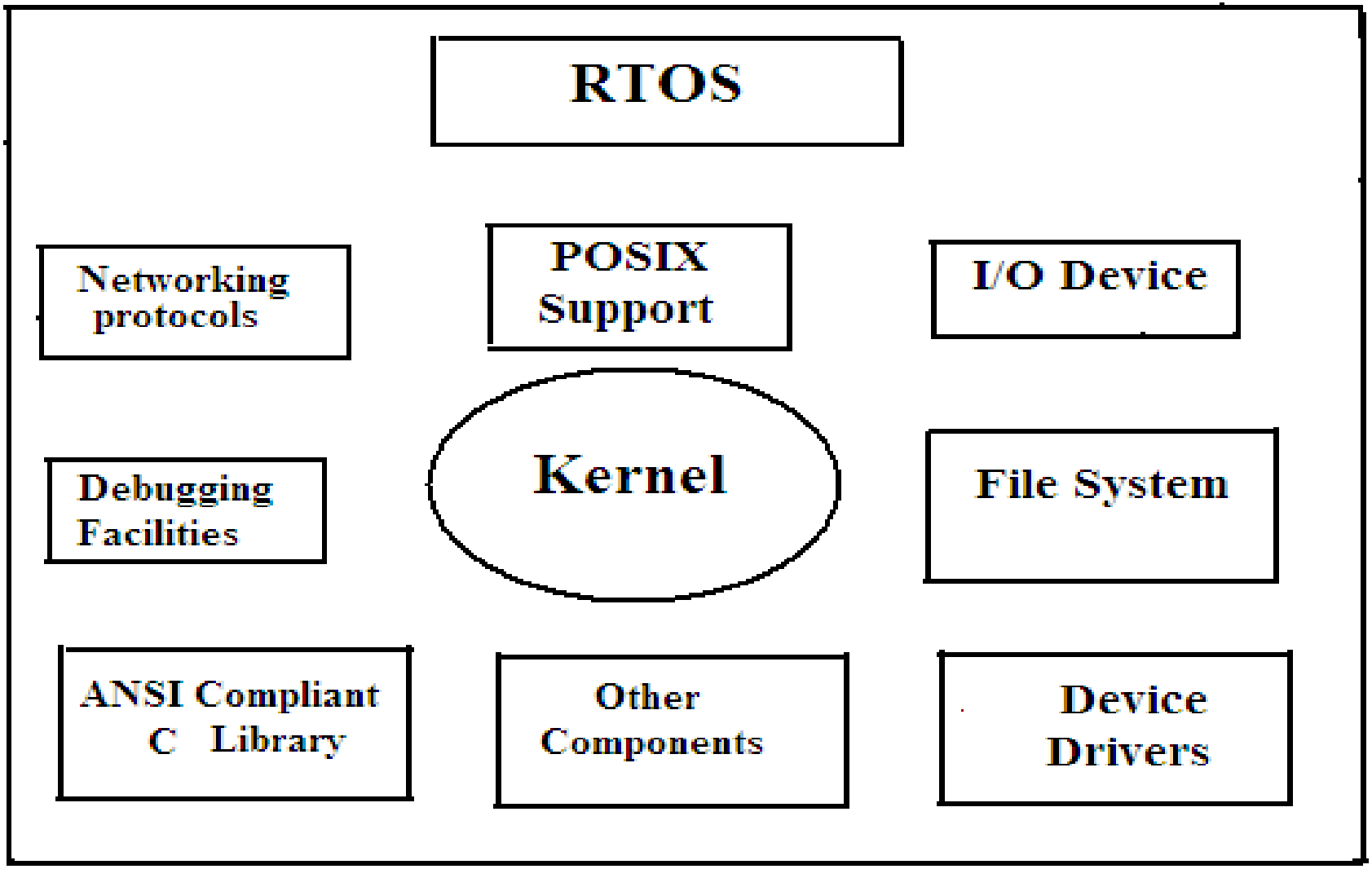
- Multithreading and preemptability.
- Thread Priority.
- Inter Task Communication & Synchronization.
- Priority Inheritance.
- Short Latencies.

RTOS Basics .....

Basic functions of RTOS:

- Time management
  - A high resolution hardware timer is programmed to interrupt the processor at fixed rate (Time interrupt)
  - Each time interrupt is called a system tick (time resolution)
- Task management
  - Task creation: create a newTCB
  - Task termination: remove the TCB
  - Change Priority: modify the TCB
  - State-inquiry: read the TCB
- Interrupt handling
  - ISR
- Memory management
  - No virtual memory for hard RT tasks, Many embedded RTS do not have memory protection
- Exception handling (important)
  - missing deadline, running out of memory, timeouts, deadlocks, divide by zero, etc.
- Task synchronization
  - Semaphore, Mutex, Avoid priority inversion
- Task scheduling
  - Priorities (HPF), Execution times (SCF), Deadlines (EDF), Arrival times (FIFO)

RTOS Basics .....



General Architecture of RTOS

RTOS Basics .....

DIFFERENCES BETWEEN RTOS and GENERAL PURPOSE OS

The basic differences are Determinism, Task scheduling and Preempting

- **Determinism:** The key difference between general-computing operating systems and real-time operating systems is the deterministic timing behavior in the real-time operating systems. "Deterministic" timing means that OS consume only known and expected amounts of time.
- **Task Scheduling:** General purpose operating systems are optimized to run a variety of applications and processes simultaneously, thereby ensuring that all tasks receive at least some processing time. RTOS uses priority-based preemptive scheduling, which allows high-priority threads to meet their deadlines consistently. The scheduling in RTOS is time based. In case of General purpose OS, scheduling is process based.
- **Pre-Emptive and Non-Pre-Emptive:** In a normal operating system, if a task is running, it will continue to run until its completion. It cannot be stopped by the OS in the middle due to any reason. Such concept is known as non-preemptive. In real time OS, a running task can be stopped due to a high priority task at any time with-out the willing of present running task. This is known as pre-emptiveness.

RTOS Basics .....

REAL TIME OPERATING SYSTEM….. Popular RTOSs:

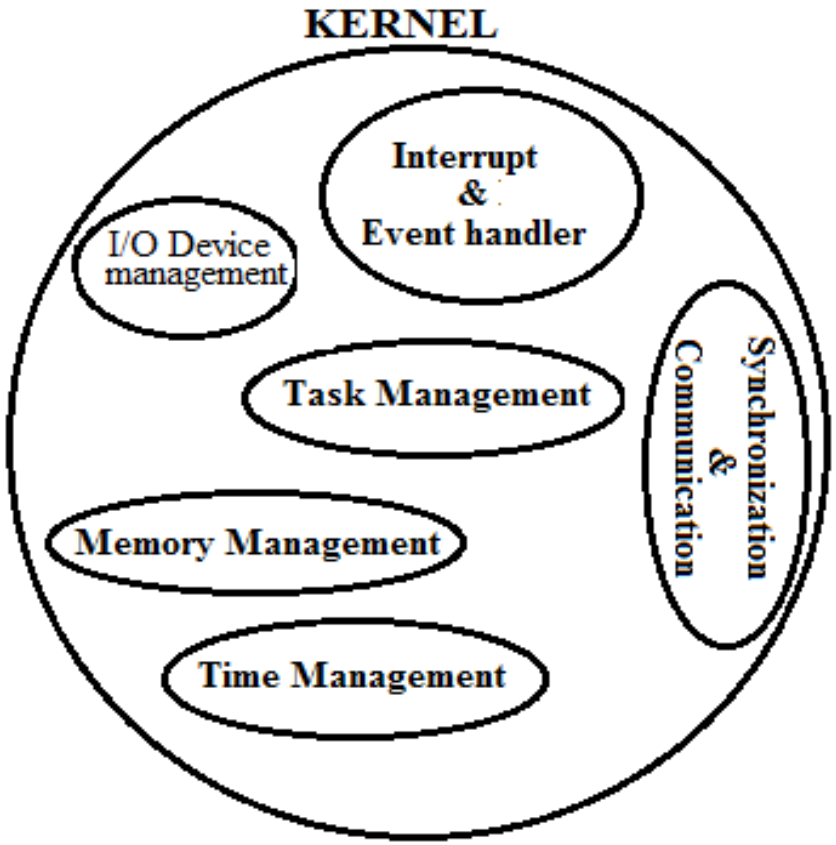
RTOS	Applications/Features
Windows CE	▪ Used small foot print mobile and connected devices ▪ Supported by ARM, MIPS & x86 architectures
LynxOS	o Complex, hard real-time applications o POSIX-compatible, multiprocess, multithreaded OS. o Supported by x86, ARM, PowerPC architectures
VxWorks	▪ Most widely adopted RTOS in the embedded industry. ▪ Used in famous NASA rover robots ▪ Certified by several agencies and for real time systems, reliability and security-critical applications.
µC/OS-II	o Ported to more than a hundred architectures including x86, mainly used in microcontrollers with low resources. o Certified by rigorous standards, such as RTCADO-178B
QNX Neutrino	▪ Most traditional RTOS in the market, Powerful hard RTOS ▪ Microkernel architecture; completely compatible with the POSIX ▪ Certified by FAADO-278 and MIL-STD-1553 standards.
Symbian	o Designed for Smartphones o Supported by ARM, x86 architecture
VRTX	▪ Traditional board based embedded systems and SoC architectures ▪ Supported by ARM, MIPS, PowerPC & other RISC architectures

Architecture of Kernel

The kernel is the core of an operating system. It is a piece of software responsible for providing secure access to the system's hardware and to running the programs.

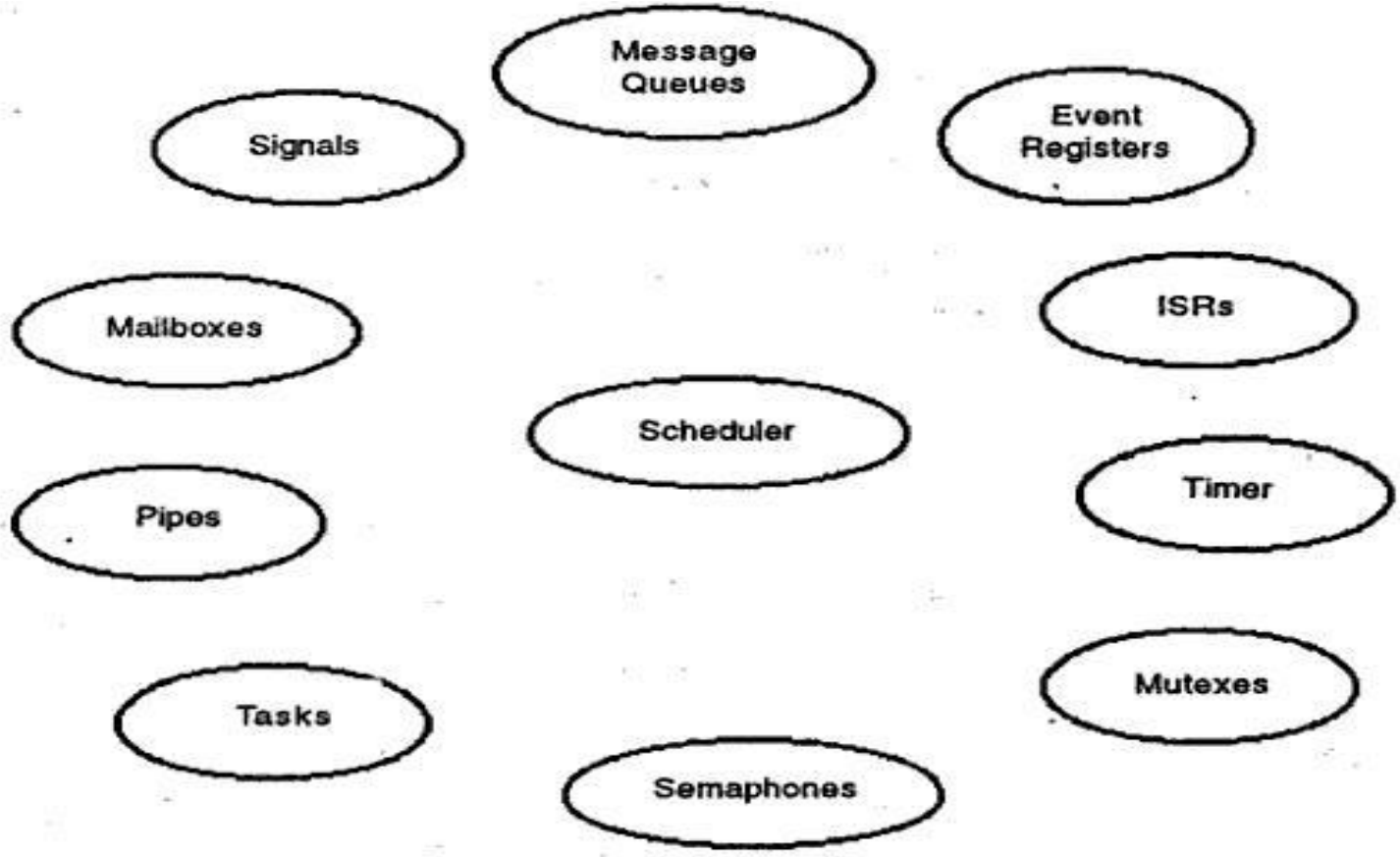
Kernel is common to every operating system either a real time or non-real time. The major difference lies in its architecture.
Since there are many programs, and hardware access is limited, the kernel also decides when and how long a program should run. This is called scheduling.

Kernels has various functions such as file management, data transfer between the file system, hardware management, memory management and also the control of CPU time. The kernel also handles the Interrupts.



Architecture of Kernel..... Kernel Objects

The various kernel objects are Tasks, Task Scheduler, Interrupt Service Routines, Semaphores, Mutexes, Mailboxes, Message Queues, Pipes, Event Registers, Signals and Timers



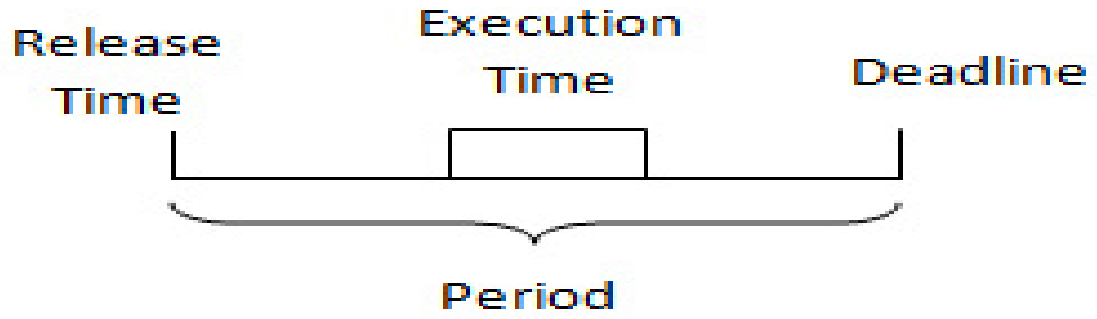
Kernel Objects

TASKS AND TASK STATES

Task

A task is a basic unit or atomic unit of execution that can be scheduled by an RTOS to use the system resources like CPU, Memory, I/O devices etc. It starts with reading of the input data and of the internal state of the task, and terminates with the production of the results and updating the internal state. The control signal that initiates the execution of a task is provided by the operating system.

In RTOS, The application is decomposed into small, schedulable, and sequential program units known as "Task", a basic unit of execution and is governed by **three time-critical properties**; release time, deadline and execution time. Release time refers to the point in time from which the task can be executed. Deadline is the point in time by which the task must complete. Execution time denotes the time the task takes to execute.



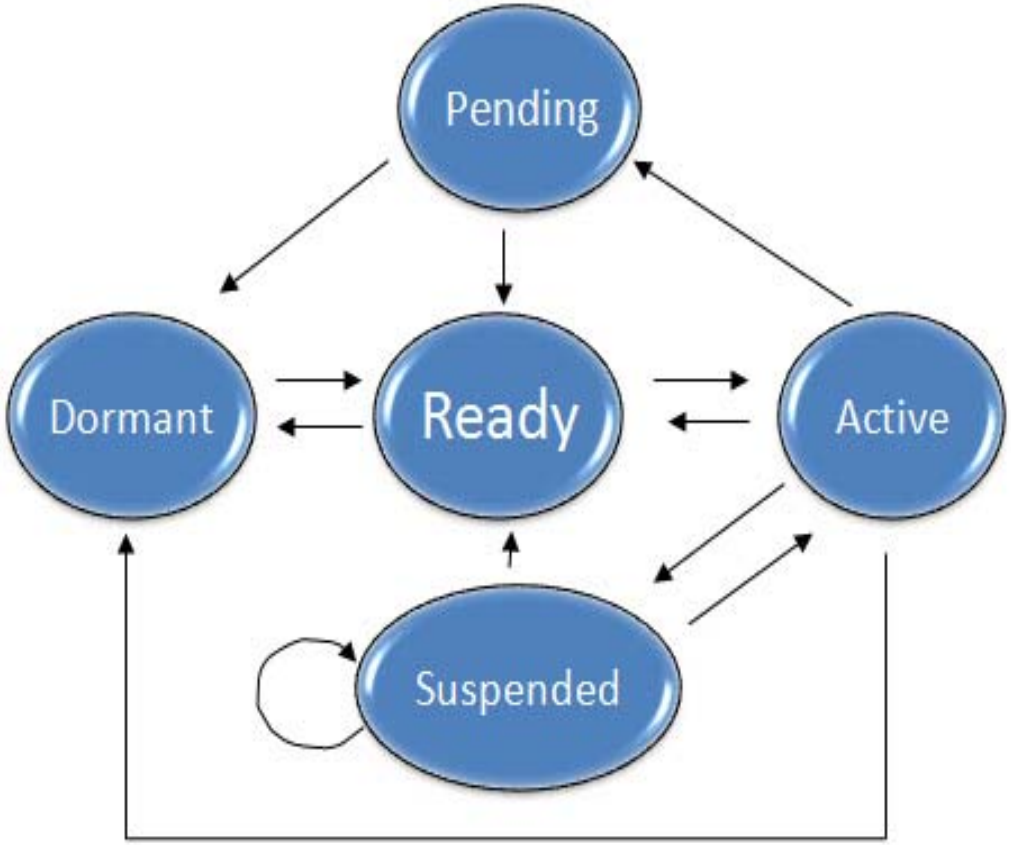
Tasks and Task States…..

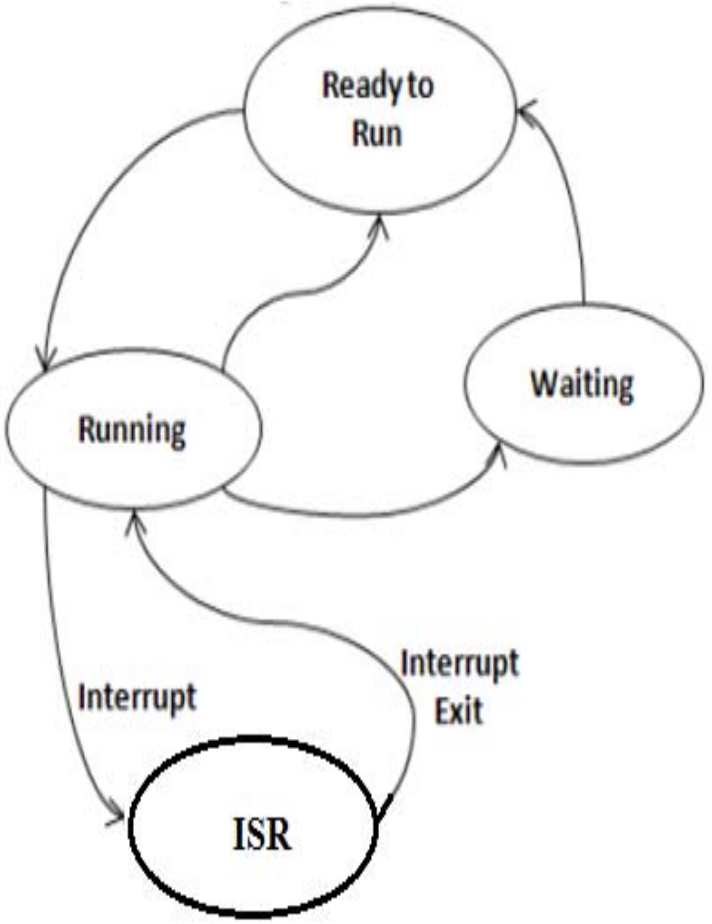
There are two types of tasks.
(i)Simple Task(S-Task) and (ii) Complex Task(C-Task).

- Simple Task (S-task): A simple task is one which has no synchronization point i.e. , whenever an S-task is started, it continues until its termination point is reached. Because an S-task cannot be blocked within the body of the task the execution time of an S-task is not directly dependent on the progress of the other tasks in the node. S-task is mainly used for single user systems.

- Complex Task (C-Task): A task is called a complex task (C-Task) if it contains a blocking synchronization statement (e.g. , a semaphore operation "wait") within the task body. Such a "wait" operation may be required because the task must wait until a condition outside the task is satisfied, e.g. , until another task has finished updating a common data structure, or until input from a terminal has arrived.

Tasks and Task States…….





Task States

Task States

Each task may exist in following states

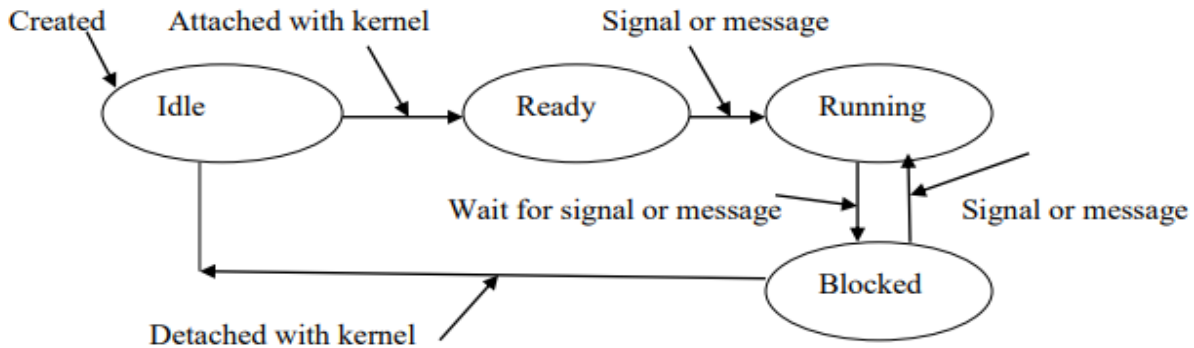
- Dormant : Task doesn't require computer time ()
- Ready: Task is ready to go active state, waiting processor time
- Active: Task is running
- Suspended: Task put on hold temporarily
- Pending: Task waiting for resource.

Tasks and Task States…..

When a task is first created, it is in the dormant task. When it is added to RTOS for scheduling, it is a ready task. If the input or a resource is not available, the task gets blocked.

During the execution of an application program, individual tasks are continuously changing from one state to another. However, only one task is in the running mode (i. e. given CPU control) at any point of the execution. In the process where CPU control is change from one task to another, context of the to-be-suspended task will be saved while context of the to-be-executed task will be retrieved, the process referred to as context switching.

If no task is ready to run and all of the tasks are blocked, the RTOS will usually run the Idle Task. An Idle Task does nothing. The idle task has the lowest priority.



Task j

Tasks and Task States…..

A task object is defined by the following set of components:

- Task Control block: Task uses TCBs to remember its context.
  TCBs are data structures residing in RAM, accessible only by RTOS
- Task Stack: These reside in RAM, accessible by stack pointer.
- Task Routine: Program code residing in ROM.