Real Time operating system

Self Introduction

- Appavuraj Murugavel
  - Core Discipline Software – Platform Integration team – Instrumentation Display.
  - Working in Continental Automotive Singapore Pte Ltd, Since September 2008
  - Expert in Software Platform Technologies.
  - Interests – Shows, Travelling.. etc
  - Today, we will have basic introduction of real time operating system in embedded systems.

Real Time Operating System
What does it mean?

It is a combination of the following two terms as defined below

Real Time + Operating System

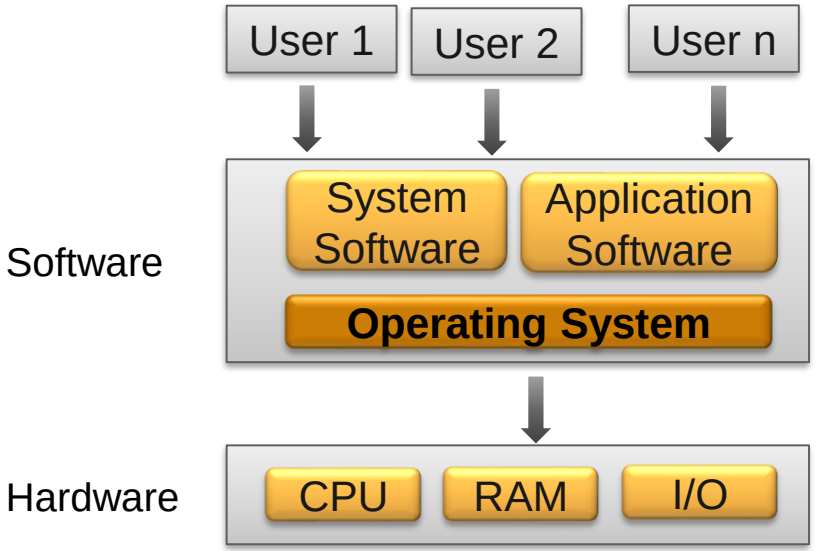
We will now have a closer look at both the terms.

The set of programs which manage the hardware resources, so that the operation is performed precisely.

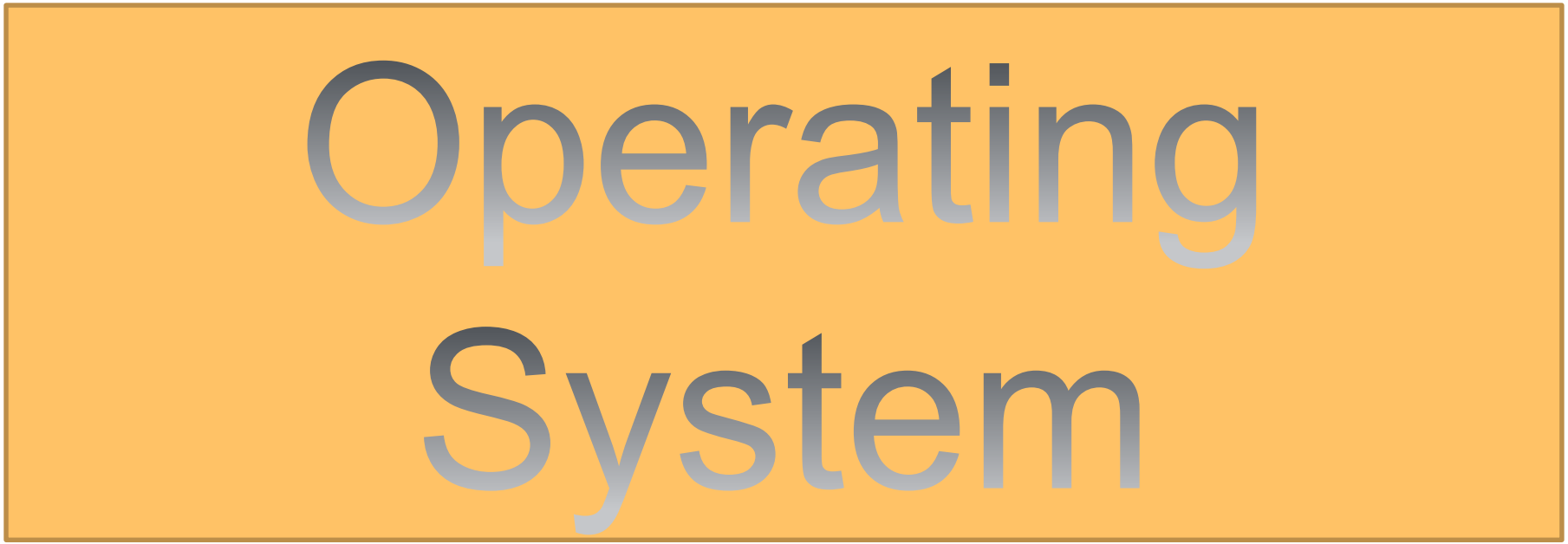
A set of programs which are used to manage all the hardware resources of a computer.

What is an Operating System

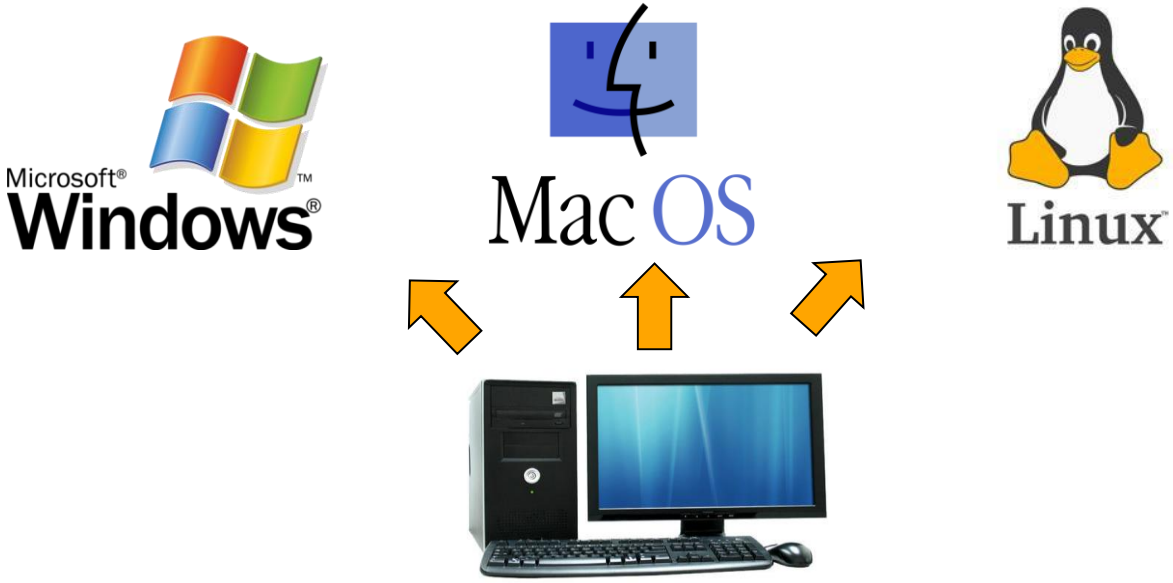
- Operating system serves as a bridge between user and computer hardware.
- Without this operating system, an user cannot run an application software.
- It is used for executing user programs and solving user problems easier.



Basics of Operating System
Components and Functionality



Examples of Operating Systems



What is a Real time system

- System that need to respond to external event in guaranteed time.
- Real time system depends of the factors as below
  - If there is a slow response, it is faulty.
  - Response time has to be quick.
- Some examples of Real time systems are shown on the other side

Types of Real Time System

- Hard Real Time Systems
  - Missing a deadline has catastrophic result to the system.
  - Example : nuclear systems, pacemakers, avionics..etc
- Firm Real Time Systems
  - Missing a deadline will cause unacceptable reduction in quality
  - Example : Mobile Phone
- Soft Real Time Systems
  - Missing a deadline will cause acceptable quality reduction.
  - Example : Web Browsing

Real Time Operating System

- An Operating System with the necessary features which supports for Real time System use case.
- Some of the examples of RTOS are listed on the other side.

Real Time Operating System - Characteristics

Predictability

- The RTOS used in this case needs to be predictable to a certain degree. The term deterministic describes RTOS with predictable behavior, in which the completion of operating system calls occurs within known timeframe

Performance

- Since the application user need to perform its activity faster and less time spend. So hence in this case the throughput of RTOS in terms of time spent and resource has to be optimal.

Compactness

- The design requirements for the application system limits the system memory used, resulting in limiting the size of the operating system to the optimal size and application module size too.

Reliability

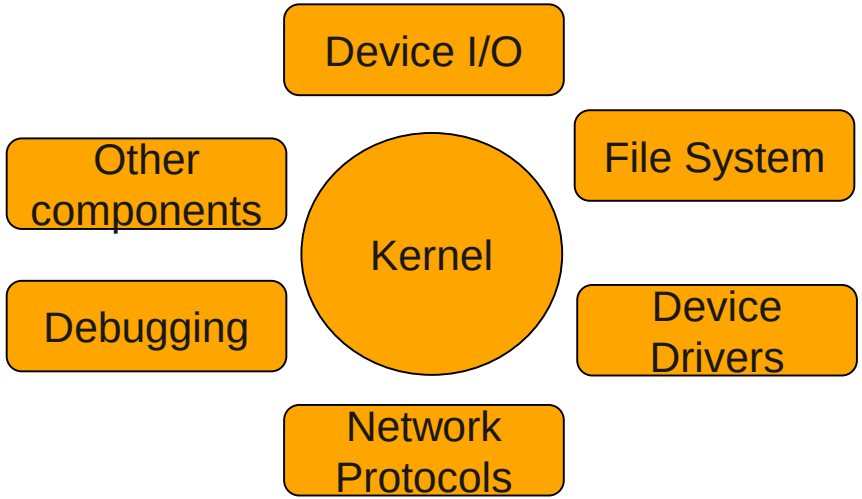
- Depending on the application, the system might need to operate for long periods without human intervention. So hence the operating system which is also used in the software system has to more reliable.

Scalability

- Depending on how much functionality is required, an RTOS should be capable of adding or deleting modular based on the application use case.

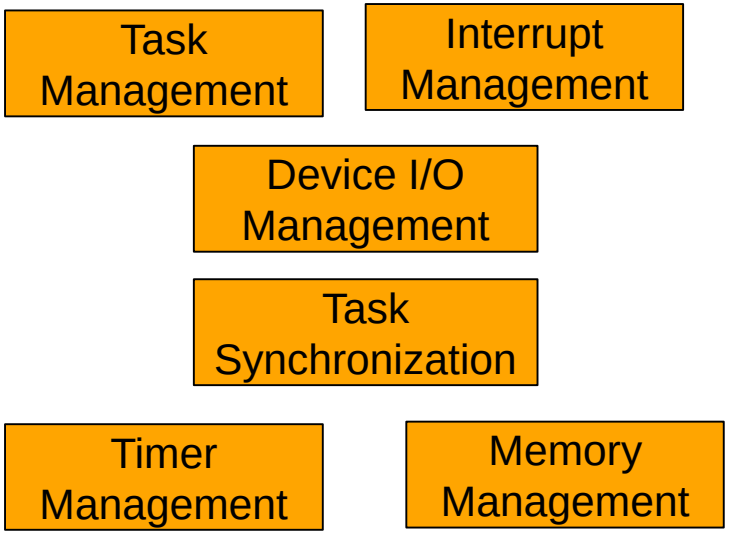
RTOS
Architectural components

- The heart of RTOS is the Kernel component, which perform Task, Interrupt, Scheduler, Memory management operations.
- RTOS can be build only with Kernel, but based on the use case the other components as indicated can be added.



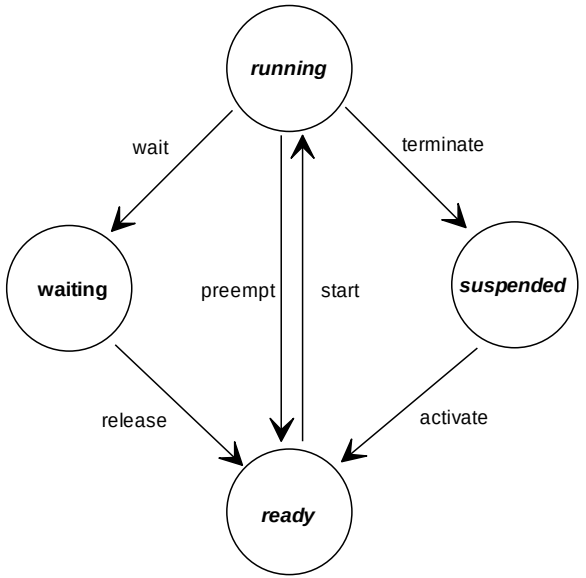
RTOS
Kernel Elements

- Kernel is the core part of RTOS system, Its service include management of Task, Interrupt, Timer, Memory and devices.
- The categories of Kernel systems are as follows
  - **Monolithic Kernel:** Perform all basic system services in kernel space.
  - **Micro Kernel:** Perform process management and I/O control in kernel space and all other system services in user space.
  - **Hybrid Kernel:** It's a combination of both monolithic and micro kernel.



Task Management

- Tasks are CPU threads which are aperiodic (Event Triggered) and Periodic (Time triggered) executing with its own stack and stack variables.
- Each task has four states as below
  - **suspended**
    - State, where the task is in passive state and can be activated.
    - When does this happen?
  - **ready**
    - Waits for the Scheduler to start the execution of the task.
  - **running**
    - Execution of the CPU thread.
  - **Waiting**
    - task cannot continue execution because it has to *wait* for at least one event

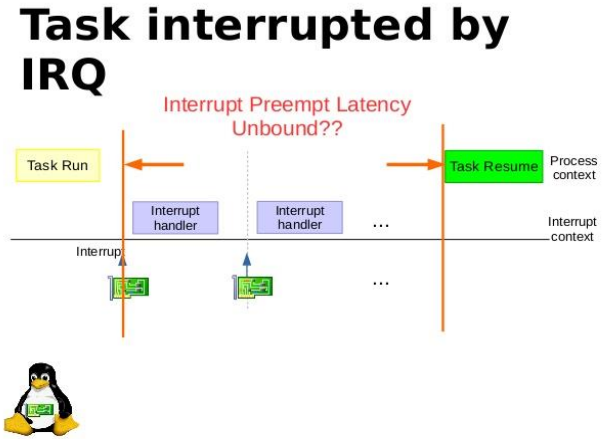


Scheduler Mechanism for context switch

- Scheduler is used to decide which ready state task is scheduled to execute,
- The different scheduling mechanism are as below
  - **Non-Preemptive Scheduling:**
    - Every task which is scheduled is execute which be running until is completed.,
  - **Round Robin Scheduling:**
    - Each task will have a fixed time slot, and runs until it is completed.
  - **Preemptive Scheduling:**
    - Each task is given a fixed unique priority level and the scheduler will schedule the task to running based on the priority of the task. Any task can be interrupted by any other higher priority task.

Interrupt Management

- When a interrupt occurs in system, the Interrupt handler is responsible to handle the interrupt by calling the defined function for the interrupt occurred based on the **interrupt vector table** addressing.
- Current running task will be pre-empted automatically when a Interrupt has occurred and the ISR (Interrupt Service Routine), will start to execute.
- After completion of the interrupt, the pre-empted task will be continued back from the same location of interruption.



Task Synchronization

- Task Synchronization
  - In a multitasking environment, you sometimes need to synchronize the order of operations between two or more tasks and ISRs. For example, some tasks may need the results of another task before executing. Depending on the execution of the process order, the system could result in Deadlock, Priority Inversion and Task Starvation problems.
- Semaphore
  - Semaphore is a signal between tasks / interrupts to synchronize the execution of the task order without any additional data. The most common type of semaphores are binary semaphore and counting semaphore.

Device I/O Management

- RTOS generally provides large number of APIs to support diverse hardware device drivers.

Timer Management

- RTOS uses a hardware system timer tick in order to perform some of the cyclic operations.
- Alarms are triggered based on the system tick count and the necessary functionalities are performed based on the user request.

Memory Management

- RTOS is responsible to perform memory management for the two memory areas – Stack and Heap.
- Stack is used during context switching and Heap is for dynamic memory allocation from a task.

Difference between RTOS and GPOS

- Real Time Operating System
  - Deterministic: no random execution pattern.
  - Predictable response time.
  - Time Bound
  - Preemptive Kernel Scheduler.
- Examples : VxWorks, OSEK, QNX

- General Purpose Operating System
  - Dynamic Memory mapping.
  - Random execution patterns.
  - Response time is not Guaranteed.
- Examples : Windows OS, MacOS, Linux.