UNIT 5

RTOS
REAL TIME OPERATING SYSTEM

Introduction

- Embedded system: used to perform a specific task and a combination of hardware and software.

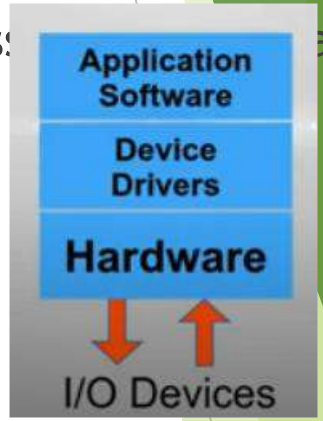
  It process the input and controls the output

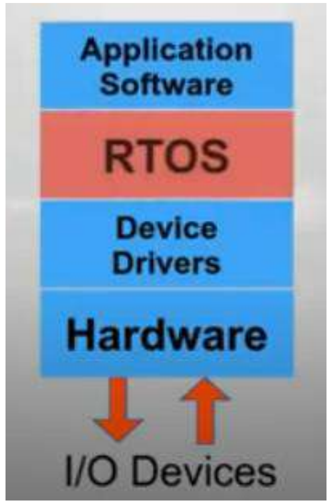
- Real time embedded system: If it has time constraint. Single proces ation software

- Real time operating system:

Complex system and require to

manage the hardware resources.



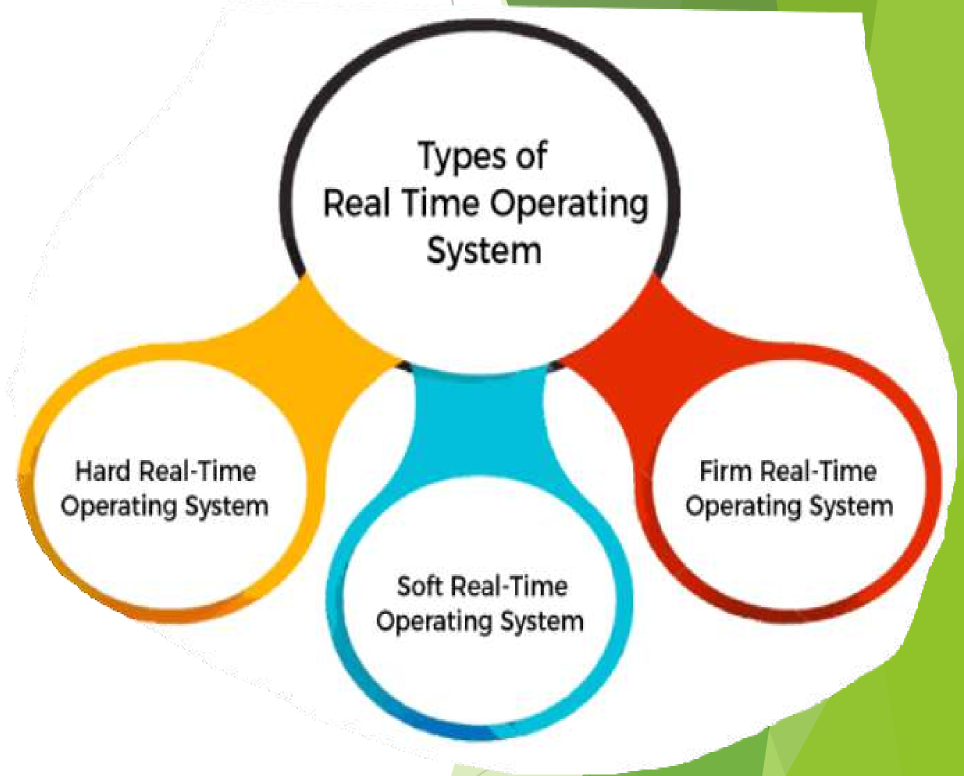


Requirements on RTOS

- Operating System(OS) is **a software that manages and handles hardware and software resources** of a computing device
- A real-time operating system (RTOS) is a specialized operating system that manages tasks with strict time constraints.
- Deterministic system calls
- Responsiveness
- Fast process/thread switch
- Fast interrupt response
- Support for concurrency and real-time
- Multi-tasking
- Real-time
- synchronization
- Scheduling, many priority

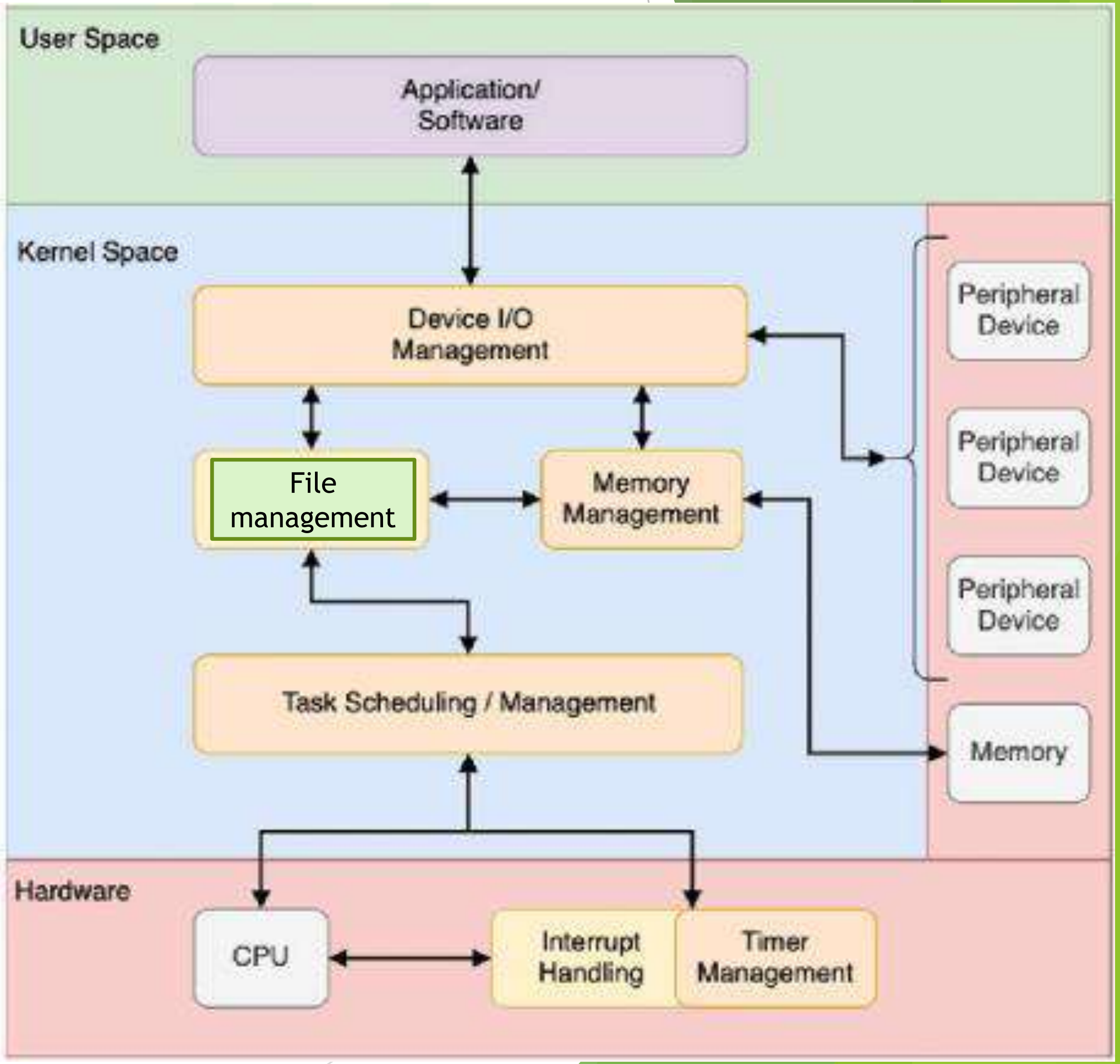
Types of RTOS

- A hard real-time operating system is used when we need to complete tasks by a given deadline. If the task is not completed on time then the system is considered to be failed.
- A soft real-time operating system is used where few delays in time duration are acceptable. That is if the given task is taking a few seconds more than the specified time then also no critical damage takes place.
- A firm real-time operating system lies between the hard and soft real-time operating system. A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete or **catastrophic** system failure. However, unlike a hard real-time task, even if a firm real-time task is not completed within its deadline, the system doesn't fail but the late results are merely discarded.
- **For example**, in Video Conferencing


Types of Real Time Operating System
Hard Real-Time Operating System
Soft Real-Time Operating System
Firm Real-Time Operating System

Architecture of RTOS

- The operating system acts as a bridge between the user applications/tasks and the underlying system resources through a set of system functionalities and services.
- The OS manages the system resources and makes them available to the user applications/tasks on a need basis.
- The primary functions of an operating system is
  - Make the system convenient to use
  - Organize and manage the system resources efficiently and correctly



Architecture of RTOS

- **The Kernel:** It is the core of the operating system and is responsible for managing the system resources and the communication among the hardware and other system services. Kernel acts as the abstraction layer between system resources and user applications. Kernel contains a set of system libraries and services.
- Task scheduling/Management: also called Process management, it deals with managing the processes/tasks. It includes setting up the memory space for the process, loading the process's code into the memory space, allocating system resources, scheduling and managing the execution of the process, Inter Process Communication and synchronisation, process termination/ deletion, etc.
- Primary Memory Management: The term primary memory refers to the volatile memory (RAM) where processes are loaded and variables and shared data associated with each process are stored.

Architecture of RTOS

- The Memory Management Unit (MMU) of the kernel is responsible for- Keeping track of which part of the memory area is currently used by which process and Allocating and De-allocating memory space on a need basis (Dynamic memory allocation).

- File system management: The file operation is a useful service provided by the OS. The file system management service of Kernel is responsible for

• The creation, deletion and alteration of files

• Creation, deletion and alteration of directories

• Saving of files in the secondary storage memory (e.g. Hard disk storage)

• Providing automatic allocation of file space based on the amount of free space available

• Providing a flexible naming convention for the files

Architecture of RTOS

I/O System (Device) Management

- Kernel is responsible for routing the I/O requests coming from different user applications to the appropriate I/O devices of the system.
- the access to i/o is provided through a set of Application Programming Interfaces (APIs) exposed by the kernel.
- The kernel maintains a list of all the I/O devices of the system. This list may be available in advance, at the time of building the kernel. Some kernels, dynamically updates the list of available devices as and when a new device is installed
- The service 'Device Manager' of the kernel is responsible for handling all I/O device related operations. T
- he kernel talks to the I/O device through a set of low-level systems calls, which are implemented in a service, called device drivers. The device drivers are specific to a device or a class of devices.
- The Device Manager is responsible for Loading and unloading of device drivers and Exchanging information and the system specific control signals to and from the device

Architecture of RTOS

- Secondary Storage Management: The secondary storage management deals with managing the secondary storage memory devices, if any, connected to the system.
- Interrupt Handler: Kernel provides handler mechanism for all external/internal interrupts generated by the system. M
- any operating systems offer a number of addon system components/services to the kernel. Network communication, network management, user-interface graphics, timer services (delays, timeouts, etc.), error handler, database management, etc. are examples for such components/services.
- Kernel exposes the interface to the various kernel applications/services, hosted by kernel, to the user applications through a set of standard Application Programming Interfaces (APIs). User applications can avail these API calls to access the various kernel application/services.

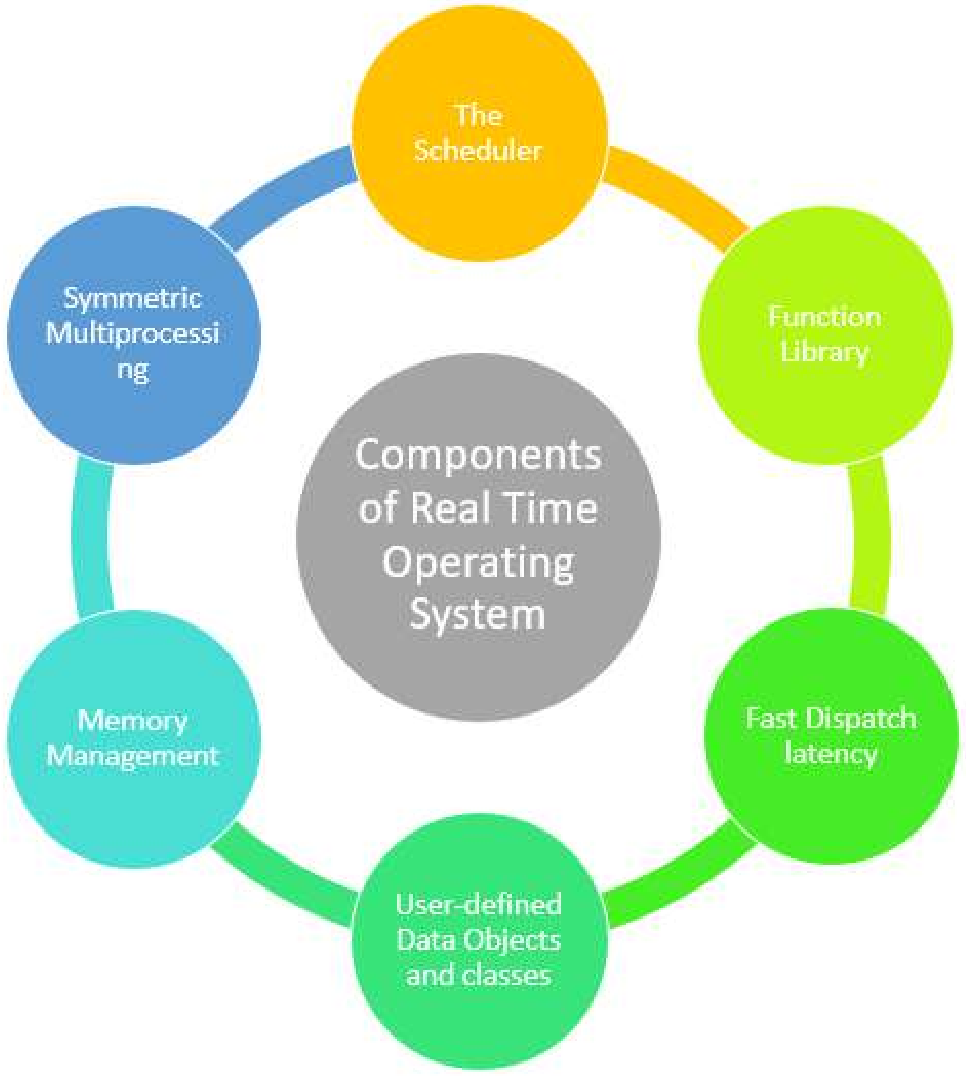
Architecture of RTOS

Kernel Space and User Space

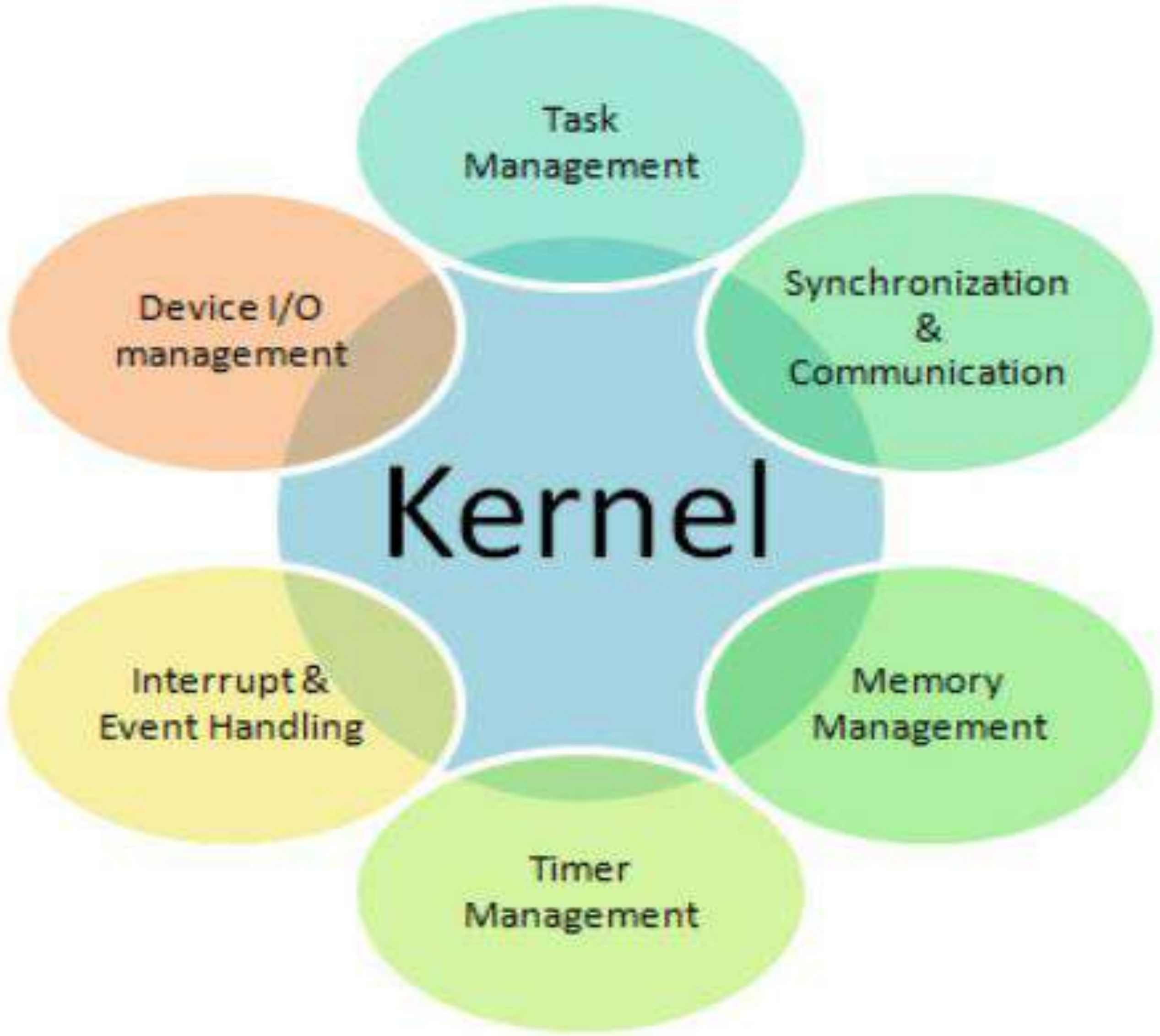
- The applications/services are classified into two categories, namely: user applications and kernel applications.
- The program code corresponding to the kernel applications/services are kept in a contiguous area (OS dependent) of primary (working) memory and is protected from the unauthorized access by user programs/applications. The memory space at which the kernel code is located is known as ‘Kernel Space’.
- All user applications are loaded to a specific area of primary memory and this memory area is referred as ‘User Space’. User space is the memory area where user applications are loaded and executed. The partitioning of memory into kernel and user space is purely Operating System dependent.

Components of RTOS

The Scheduler: Tells that in which order, the tasks can be executed which is generally based on the priority.
Symmetric Multiprocessing (SMP): It is a number of multiple different tasks that can be handled by the RTOS so that parallel processing can be done.
Memory Management: This element is needed in the system to allocate memory to every program
User-defined data objects and classes: RTOS system makes use of programming languages like C or C++, which should be organized according to their operation.
Fast dispatch latency: It is an interval between the termination of the task that can be identified by the OS and the actual time taken by the thread, which is in the ready queue, that has started processing.
Function Library: It is an important element of RTOS that acts as an interface that helps you to connect kernel and application code. This application allows to send the requests to the Kernel using a function library so that the application can give the desired results..


Components of Real Time Operating System
The Scheduler
Function Library
Fast Dispatch latency
User-defined Data Objects and classes
Memory Management
Symmetric Multiprocessi ng

Services/Functions offered by RTOS



next

RTOS SCHEDULING