SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ITT204 - MICROCONTROLLER AND EMBEDDED SYSTEMS

II YEAR/ IV SEMESTER

UNIT IV PROCESSES AND OPERATING SYSTEMS

TOPIC – Types of Operating Systems

Types of Operating Systems:

Depending on the type of kernel and kernel services, purpose and type of computing systems where the OS is deployed and the responsiveness to applications, Operating Systems are classified into

1. **General Purpose Operating System (GPOS):**
2. **Real Time Purpose Operating System (RTOS):**

1. General Purpose Operating System (GPOS):

- Operating Systems, which are deployed in general computing systems
- The kernel is more generalized and contains all the required services to execute generic applications
- Need not be deterministic in execution behavior
- May inject random delays into application software and thus cause slow responsiveness of an application at unexpected times
- Usually deployed in computing systems where deterministic behavior is not an important criterion
- Personal Computer/Desktop system is a typical example for a system where GPOSs are deployed.
- Windows XP/MS-DOS etc are examples of General Purpose Operating System

2. Real Time Purpose Operating System (RTOS):

- Operating Systems, which are deployed in embedded systems demanding real-time response
- Deterministic in execution behavior. Consumes only known amount of time for kernel applications
- Implements scheduling policies for executing the highest priority task/application always
- Implements policies and rules concerning time-critical allocation of a system's resources
- Windows CE, QNX, VxWorks , MicroC/OS-II etc are examples of Real Time Operating Systems (RTOS)

The Real Time Kernel: The kernel of a Real Time Operating System is referred as Real Time kernel. In complement to the conventional OS kernel, the Real Time kernel is highly specialized and it contains only the minimal set of services required for running the user applications/tasks. The basic functions of a Real Time kernel are

a) Task/Process management

b) Task/Process scheduling

c) Task/Process synchronization

d) Error/Exception handling

e) Memory Management

f) Interrupt handling

g) Time management

- **Real Time Kernel Task/Process Management:** Deals with setting up the memory space for the tasks, loading the task's code into the memory space, allocating system resources, setting up a Task Control Block (TCB) for the task and task/process termination/deletion. A Task Control Block (TCB) is used for holding the information corresponding to a task. TCB usually contains the following set of information

  - *Task ID:* Task Identification Number
  - *Task State:* The current state of the task. (E.g. State= 'Ready' for a task which is ready to execute)
  - *Task Type*: Task type. Indicates what is the type for this task. The task can be a hard real time or soft real time or background task.
  - *Task Priority*: Task priority (E.g. Task priority =1 for task with priority = 1)
  - *Task Context Pointer*: Context pointer. Pointer for context saving

- *Task Memory Pointers*: Pointers to the code memory, data memory and stack memory for the task
- *Task System Resource Pointers*: Pointers to system resources (semaphores, mutex etc) used by the task
- *Task Pointers:* Pointers to other TCBs (TCBs for preceding, next and waiting tasks)
- *Other Parameters* Other relevant task parameters

The parameters and implementation of the TCB is kernel dependent. The TCB parameters vary across different kernels, based on the task management implementation

The parameters and implementation of the TCB is kernel dependent. The TCB parameters vary across different kernels, based on the task management implementation

- **Task/Process Scheduling:** Deals with sharing the CPU among various tasks/processes. A kernel application called '*Scheduler*' handles the task scheduling. Scheduler is nothing but an algorithm implementation, which performs the efficient and optimal scheduling of tasks to provide a deterministic behavior.

- **Task/Process Synchronization:** Deals with synchronizing the concurrent access of a resource, which is shared across multiple tasks and the communication between various tasks.

- **Error/Exception handling:** Deals with registering and handling the errors occurred/exceptions raised during the execution of tasks. Insufficient memory, timeouts, deadlocks, deadline missing, bus error, divide by zero, unknown instruction execution etc, are examples of errors/exceptions. Errors/Exceptions can happen at the kernel level services or at task level. Deadlock is an example for kernel level exception, whereas timeout is an example for a task level exception. The OS kernel gives the information about the error in the form of a system call (API).

Hard Real-time System:

- A Real Time Operating Systems which strictly adheres to the timing constraints for a task
- A Hard Real Time system must meet the deadlines for a task without any slippage
- Missing any deadline may produce catastrophic results for Hard Real Time Systems, including permanent data lose and irrecoverable damages to the system/users
- Emphasize on the principle '*A late answer is a wrong answer*'
- Air bag control systems and Anti-lock Brake Systems (ABS) of vehicles are typical examples of Hard Real Time Systems
- As a rule of thumb, Hard Real Time Systems does not implement the virtual memory model for handling the memory. This eliminates the delay in swapping in and out the code corresponding to the task to and from the primary memory

- The presence of *Human in the loop* (*HITL*) for tasks introduces un-expected delays in the task execution. Most of the Hard Real Time Systems are automatic and does not contain a 'human in the loop'

- **Soft Real-time System:**
  - Real Time Operating Systems that does not guarantee meeting deadlines, but, offer the best effort to meet the deadline
  - Missing deadlines for tasks are acceptable if the frequency of deadline missing is within the compliance limit of the Quality of Service (QoS)
  - A Soft Real Time system emphasizes on the principle '*A late answer is an acceptable answer, but it could have done bit faster*'
  - Soft Real Time systems most often have a '*human in the loop (HITL)*'

- Automatic Teller Machine (ATM) is a typical example of Soft Real Time System. If the ATM takes a few seconds more than the ideal operation time, nothing fatal happens.
- An audio video play back system is another example of Soft Real Time system. No potential damage arises if a sample comes late by fraction of a second, for play back.

THANK YOU