REAL-TIME OPERATING SYSTEMS

CONTENTS

GENERAL PURPOSE OPERATING SYSTEM

- An interface between users and hardware
- Controlling and allocating memory
- Controlling input and output devices
- Managing file systems
- Facilitating networking

NON-REAL-TIME SYSTEMS

- Non-Real-Time systems are the operating systems most often used
- No guaranteed worst case scheduling jitter
- System load may result in delayed interrupt response
- System response is strongly load dependent
- System timing is a unmanaged resource

WHAT IS A RTOS ??

- RTOS is a pre-emptive multitasking operating system intended for real-time applications
- Predictable OS timing behavior
- Able to determine task's completion time
- A system of priority inheritance has to exist
- Guarantees task completion at a set deadline.

TYPES OF RTOS

- Soft Real-Time system
- Hard Real-Time system

Soft Real-Time system

- The soft real-time definition allows for frequently missed deadlines
- If the system fails to meet the deadline, possibly more than once ,the system is not considered to have failed
- Example : Multimedia streaming , Video games

HARD REAL-TIME SYSTEM

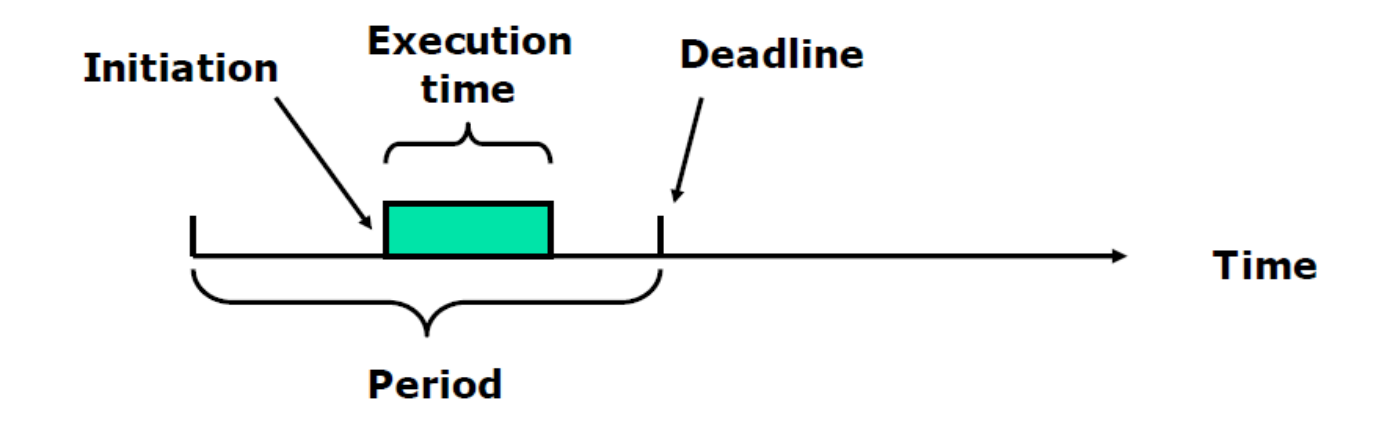
- A hard real-time system guarantees that real-time tasks be completed within their required deadlines
- Failure to meet a single deadline may lead to a critical system failure
- Examples: air traffic control , vehicle subsystems control, medical systems

BASIC FUNCTIONS OF RTOS KERNEL

- Task Management
- Interrupt handling
- Memory management
- Exception handling
- Task synchronization
- Task scheduling
- Time management

TASK MANAGEMENT

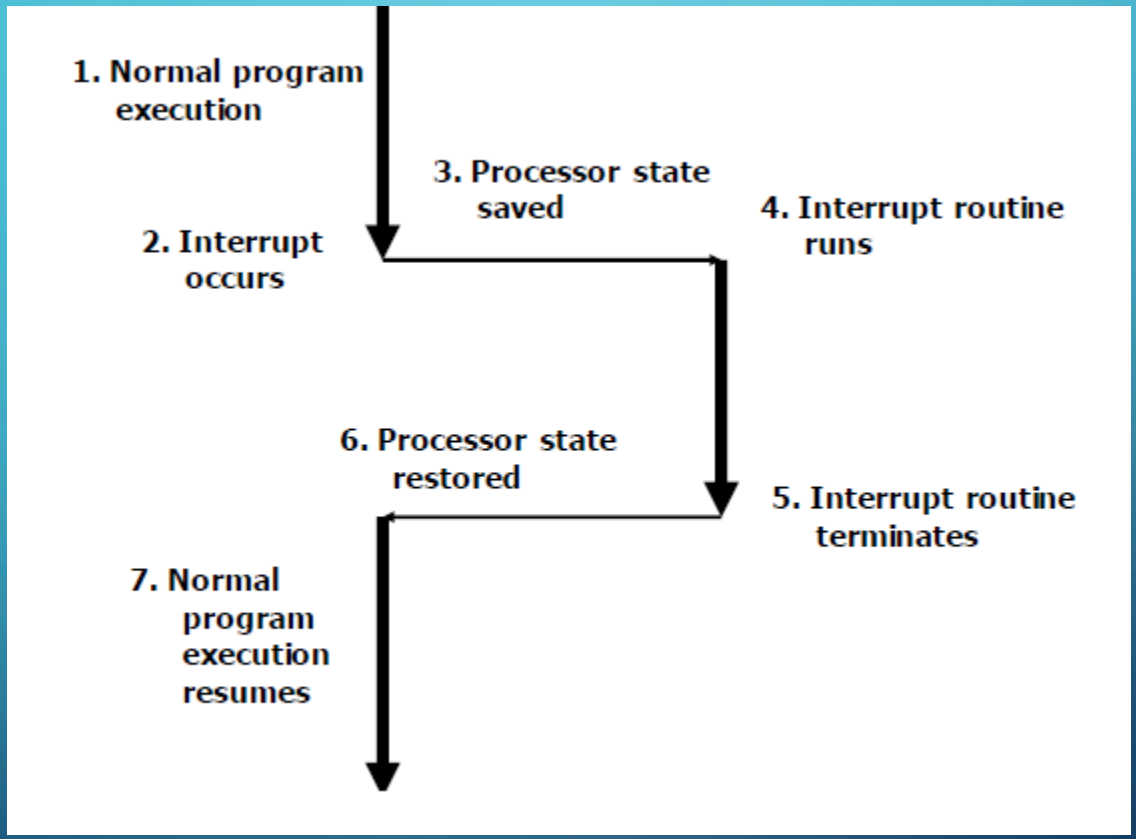
- Tasks are implemented as threads in RTOS
- Have timing constraints for tasks
- Each task a triplet: (execution time, period, deadline)
- Can be initiated any time during the period



TASK STATES

- **Idle** : task has no need for computer time
- **Ready** : task is ready to go active, but waiting for processor time
- **Running** : task is executing associated activities
- **Waiting** : task put on temporary hold to allow lower priority task chance to execute
- **suspended**: task is waiting for resource

INTERRUPT HANDLING



INTERRUPT HANDLING

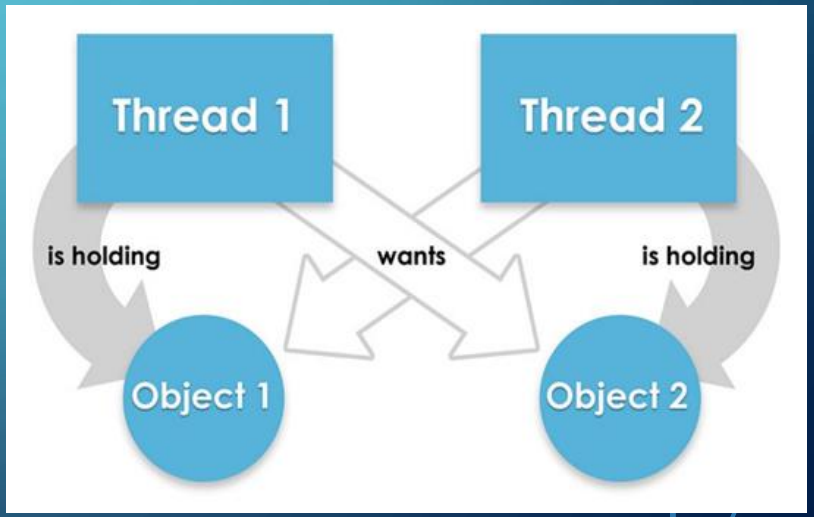
- Types of interrupts
  - **Asynchronous or hardware interrupt**
  - **Synchronous or software interrupt**
- Very low Interrupt latency
- The ISR of a lower-priority interrupt may be blocked by the ISR of a high-priority

MEMORY MANAGEMENT

- RTOS may disable the support to the dynamic block allocation
- When a task is created the RTOS simply returns an already initialized memory location
- when a task dies, the RTOS returns the memory location to the pool
- No virtual memory for hard RT tasks

EXCEPTION HANDLING

- Exceptions are triggered by the CPU in case of an error
- E.g. : Missing deadline, running out of memory, timeouts, deadlocks, divide by zero, etc.
  - Error at system level, e.g. deadlock
  - Error at task level, e.g. timeout

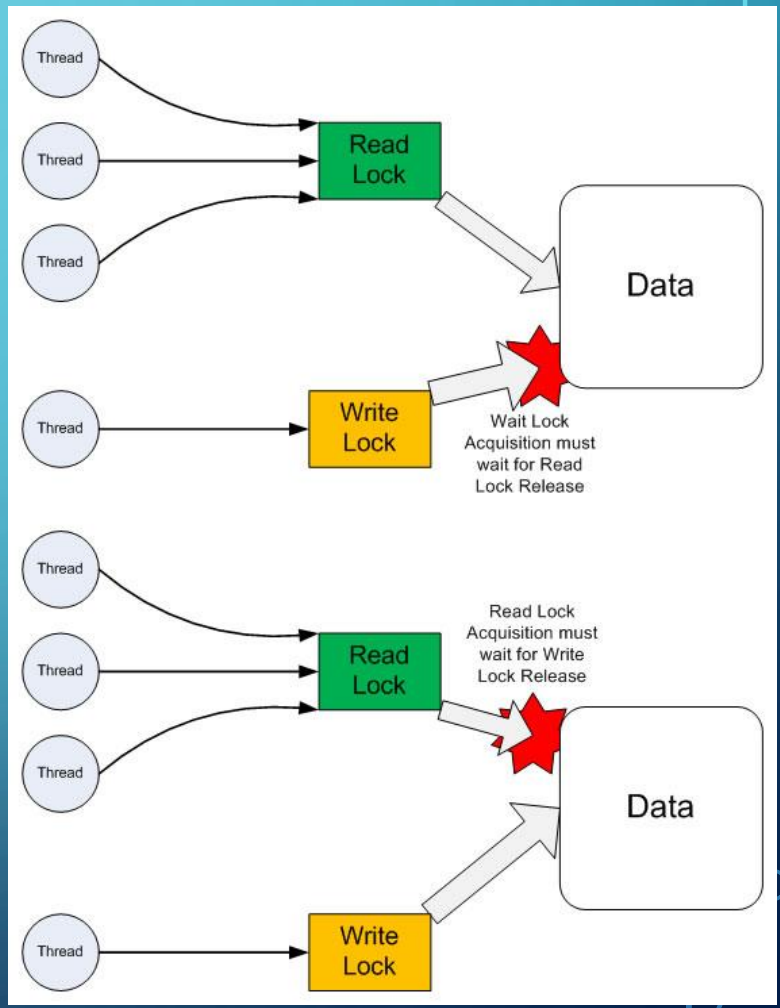


EXCEPTION HANDLING

- Standard techniques:
  - System calls with error code
  - Watch dog
  - Fault-tolerance
- Missing one possible case may result in disaster

TASK SYNCHRONIZATION

- Semaphore
- Mutex
- Spinlock
- Read/write locks



TASK SCHEDULING

- Scheduler is responsible for time-sharing of CPU among tasks
  - Priority-based Preemptive Scheduling
  - Rate Monotonic Scheduling
  - Earliest Deadline First Scheduling
  - Round robin scheduling

TASK SCHEDULING

- **Priority-based Preemptive Scheduling**
  - Assign each process a priority
  - At any time, scheduler runs highest priority process ready to run

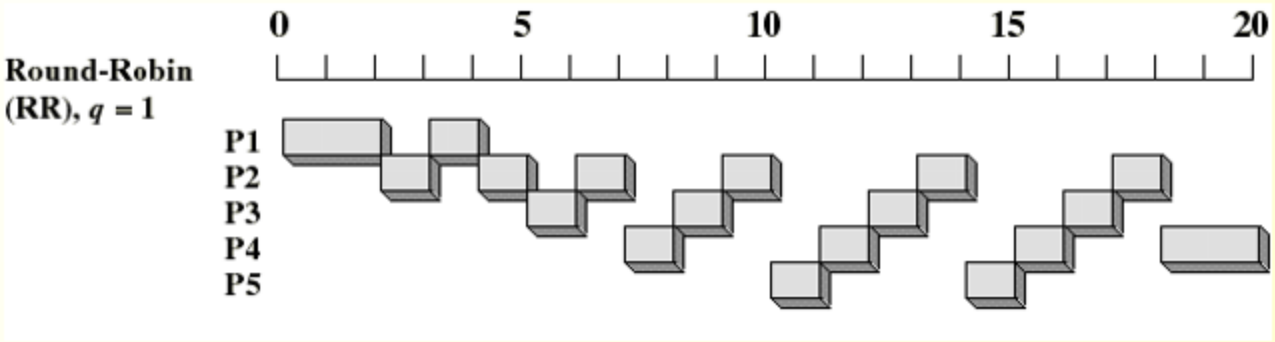
- **Rate Monotonic Scheduling**
  - A priority is assigned based on the inverse of its period
  - Shorter execution periods = higher priority
  - Longer execution periods = lower priority

TASK SCHEDULING

- **Earliest Deadline First Scheduling**
  - Priorities are assigned according to deadlines
  - The earlier the deadline, the higher the priority
  - Priorities are dynamically chosen
- **Round robin scheduling**
  - Designed for time-sharing systems
  - Jobs get the CPU for a fixed time
  - Ready queue treated as a circular buffer
  - Process may use less than a full time slice

EXAMPLE OF TASK SCHEDULING (RR)

Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



TIME MANAGEMENT

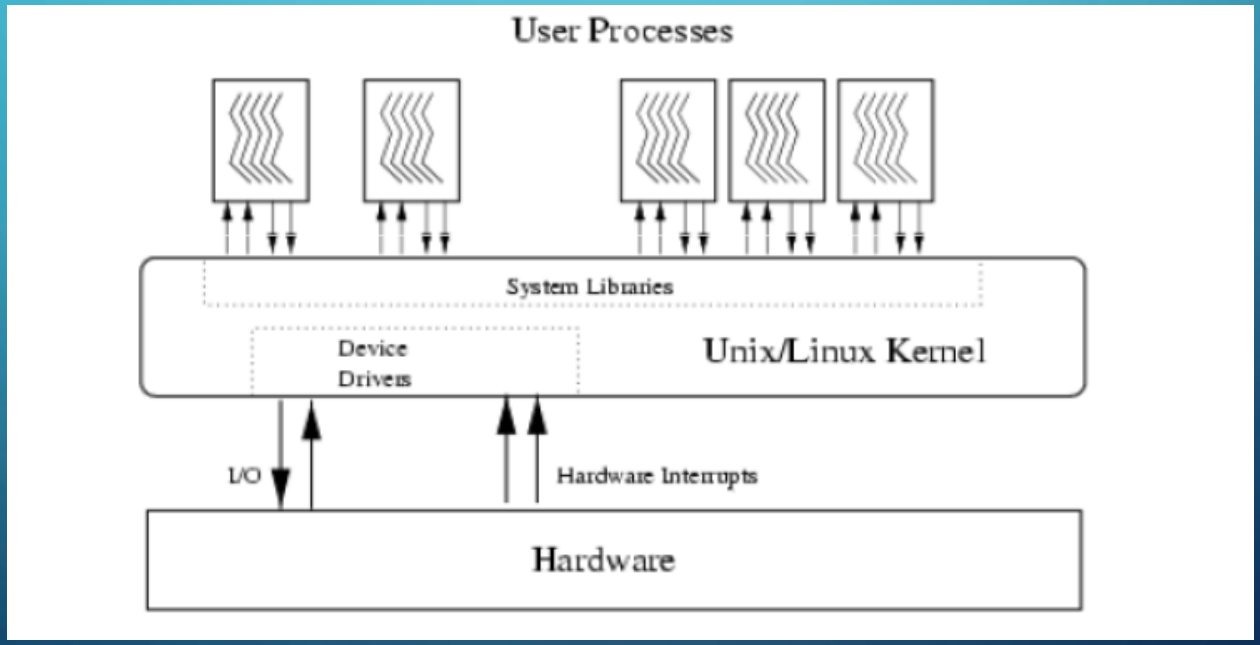
- **Time interrupt** : A high resolution hardware timer is programmed to interrupt the processor at fixed rate
- Each time interrupt is called a system tick
- The tick may be chosen according to the given task parameters

EXISTING RTOS CATEGORIES

- Priority based kernel for embbeded applications
  - VxWorks, OSE, QNX
- Real Time Extensions of existing time-sharing OS
  - Real time Linux , Real time NT
- Research RT Kernels
  - MARS, Spring

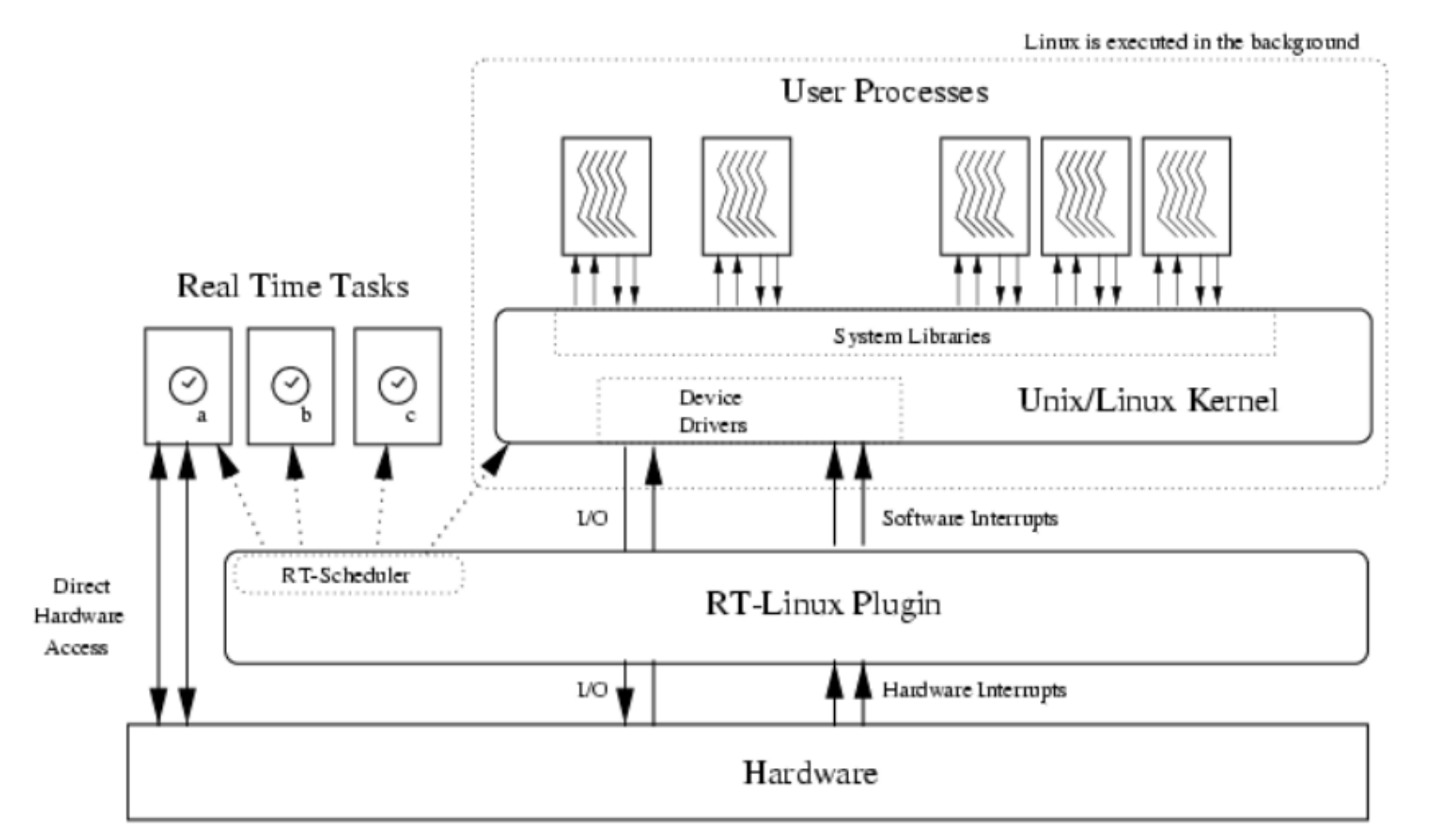
RT Linux: an example

- RT-Linux is an operating system, in which a small real-time kernel co-exists with standard Linux kernel



Non RT Kernel

RT Linux Kernel



Conclusion

- RTOS is an OS for response time controlled and event controlled processes. The processes have predicable latencies and execute by pre-emptive scheduling
- An RTOS is an OS for the systems having the hard or soft real timing constraints and deadline on the tasks

Thank you…!!!