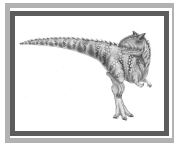


# Chapter 19: Real-Time Systems



## Chapter 19: Real-Time Systems

- System Characteristics
- Features of Real-Time Systems
- Implementing Real-Time Operating Systems
- Real-Time CPU Scheduling
- An Example: VxWorks 5.x



## Objectives

- To explain the timing requirements of real-time systems
- To distinguish between hard and soft real-time systems
- To discuss the defining characteristics of real-time systems
- To describe scheduling algorithms for hard real-time systems



## Overview of Real-Time Systems

- A **real-time system** requires that results be produced within a specified **deadline period**
- An **embedded system** is a computing device that is part of a larger system (i.e. automobile, airliner)
- A **safety-critical system** is a real-time system with catastrophic results in case of failure
- A **hard real-time system** guarantees that real-time tasks be completed within their required deadlines
- A **soft real-time system** provides priority of real-time tasks over non real-time tasks





## System Characteristics

- Single purpose
- Small size
- Inexpensively mass-produced
- Specific timing requirements

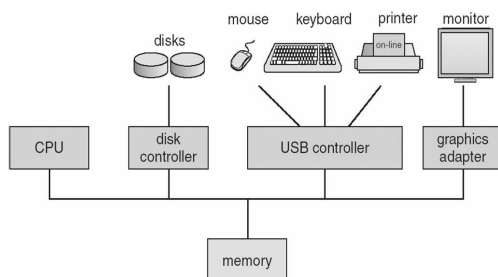


## System-on-a-Chip

- Many real-time systems are designed using system-on-a-chip (SOC) strategy
- SOC allows the CPU, memory, memory-management unit, and attached peripheral ports (i.e. USB) to be contained in a single integrated circuit



## Bus-Oriented System



## Features of Real-Time Kernels

- Most real-time systems do not provide the features found in a standard desktop system
- Reasons include
  - Real-time systems are typically single-purpose
  - Real-time systems often do not require interfacing with a user
  - Features found in a desktop PC require more substantial hardware that what is typically available in a real-time system



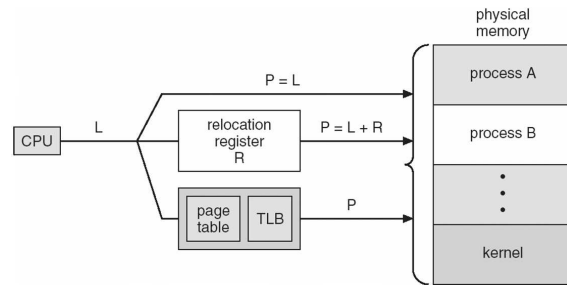


## Virtual Memory in Real-Time Systems

- Address translation may occur via:
  - (1) Real-addressing mode where programs generate actual addresses
  - (2) Relocation register mode
  - (3) Implementing full virtual memory



## Address Translation



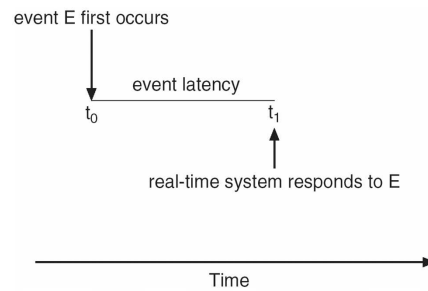
## Implementing Real-Time Systems

- In general, real-time operating systems must provide:
  - (1) Preemptive, priority-based scheduling
  - (2) Preemptive kernels
  - (3) Latency must be minimized



## Minimizing Latency

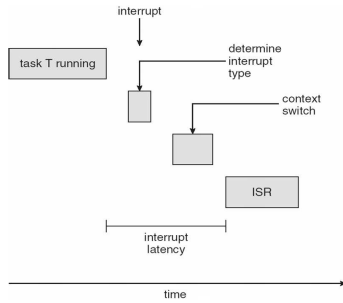
- **Event latency** is the amount of time from when an event occurs to when it is serviced.





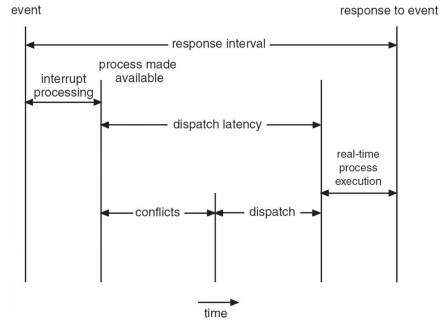
## Interrupt Latency

- Interrupt latency is the period of time from when an interrupt arrives at the CPU to when it is serviced



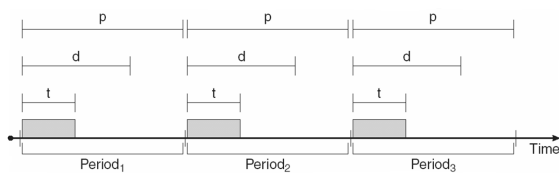
## Dispatch Latency

- Dispatch latency is the amount of time required for the scheduler to stop one process and start another

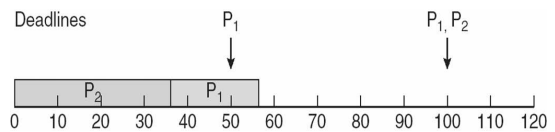


## Real-Time CPU Scheduling

- Periodic processes require the CPU at specified intervals (periods)
- $p$  is the duration of the period
- $d$  is the deadline by when the process must be serviced
- $t$  is the processing time



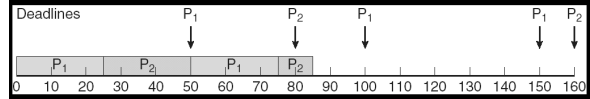
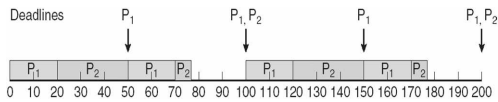
## Scheduling of tasks when $P_2$ has a higher priority than $P_1$



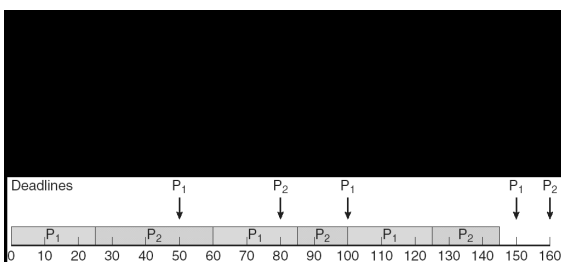


## Rate Monotonic Scheduling

- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority;
- Longer periods = lower priority
- $P_1$  is assigned a higher priority than  $P_2$ .



## Earliest Deadline First Scheduling



## Proportional Share Scheduling

- $T$  shares are allocated among all processes in the system
- An application receives  $N$  shares where  $N < T$
- This ensures each application will receive  $N / T$  of the total processor time



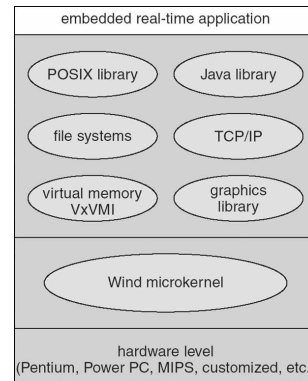


## Pthread Scheduling

- The Pthread API provides functions for managing real-time threads
- Pthreads defines two scheduling classes for real-time threads:
  - (1) SCHED\_FIFO - threads are scheduled using a FCFS strategy with a FIFO queue. There is no time-slicing for threads of equal priority
  - (2) SCHED\_RR - similar to SCHED\_FIFO except time-slicing occurs for threads of equal priority



## VxWorks 5.0



## Wind Microkernel

- The Wind microkernel provides support for the following:
  - (1) Processes and threads
  - (2) preemptive and non-preemptive round-robin scheduling
  - (3) manages interrupts (with bounded interrupt and dispatch latency times)
  - (4) shared memory and message passing interprocess communication facilities



## End of Chapter 19

