Real Time Operating Systems

Introduction

Lecturer:
Prof. William Fornaciari
Politecnico di Milano
fornacia@elet.polimi.it
www.elet.polimi.it/~fornacia

Outline

- Introduction to Real Time Systems
- The role of a RTOS
- Time constraints
- *The problem of scheduling RT activities*
- *Unix SVR4, Win2000, Linux*
- *VxWork real-time features*
- *Windows CE real-time features*

Real time systems

- Definition of real-time system
  - “A real-time system is one in which the correctness of the computation not only depends on the logical correctness of the computation, but also on the time at which the result is computed. If the timing constraints of the system are not met, system failure is said to have occurred”
- Tasks or processes attempt to control or react to external events occuring in “real time”
- RT processes must be able to keep up events
  - Hard Real Time: timing violation produces a disaster
  - Soft Real Time: timing violations are undesirable but not chatastofic

Real time systems: Examples

- Control of laboratory experiments
- Robotics
- Telecommunications (e.g. SIEMENS mobile products)
- Process control plants
- Air traffic control
- Some automotive subsystems
- Military and control systems
- …

RTOS in Real-Time System

- The RTOS is just one element of the complete real-time system
- The RTOS must provide sufficient functionality to enable the overall real-time system to meet its requirements
- RTOSs should ensure that all time critical events are handled as quickly and efficiently as possible
- RTOS are characterized by these main features
  - deterministic behaviour
  - responsiveness
  - user control
  - reliability and fail-soft operation

Determinism

- Operations are performed at fixed, predetermined times or within predetermined time intervals
- Concerned with
  - how long the operating system delays **before** acknowledging an interrupt
  - whether the system has sufficient capacity to handle all requests within required time
- RTOS should present minimum interrupt latency, typical measure
  - max delay to start ISR with highest priority
  - RT: µs ..ms
  - No RT: 10..100 ms (actually no upper bound)

Responsiveness

- It concerns with how long, after acknowledgment, it takes the operating system to **service** the interrupt
- Includes amount of time to begin execution of the ISR
  - if it is necessary a context switch, the delay is longer w.r.t. ISR executed within the context of the current process
- Includes the amount of time to perform the interrupt, depending also on hw
- Depends on the possible ISR nesting and if ISRs can be interrupted (thus the service delayed)

User Control

- Much broader in RTOS than in ordinary O.S.
- User should be able to
  - Specify paging or process swapping
  - Decide what processes must reside in main memory
  - Establish the rights of processes
  - Fine grain control over task priorities
  - Select algorithms for disks scheduling
  - …

Reliability

- In the application domain of RTOS degradation of performances may have catastrophic consequences
- Attempt either to fix the problem or minimize its effects while continuing to run
- Graceful degradation can be not enough
- **Fail-soft**: ability to fail in such a way to preserve as much data and capability is possible (e.g., multiple attempts to improve data consistency)
- **Stability**: the system adopts policies such that the most critical high priority tasks execute first, even if some needs of less important tasks can be not always met

Typical features of a RTOS

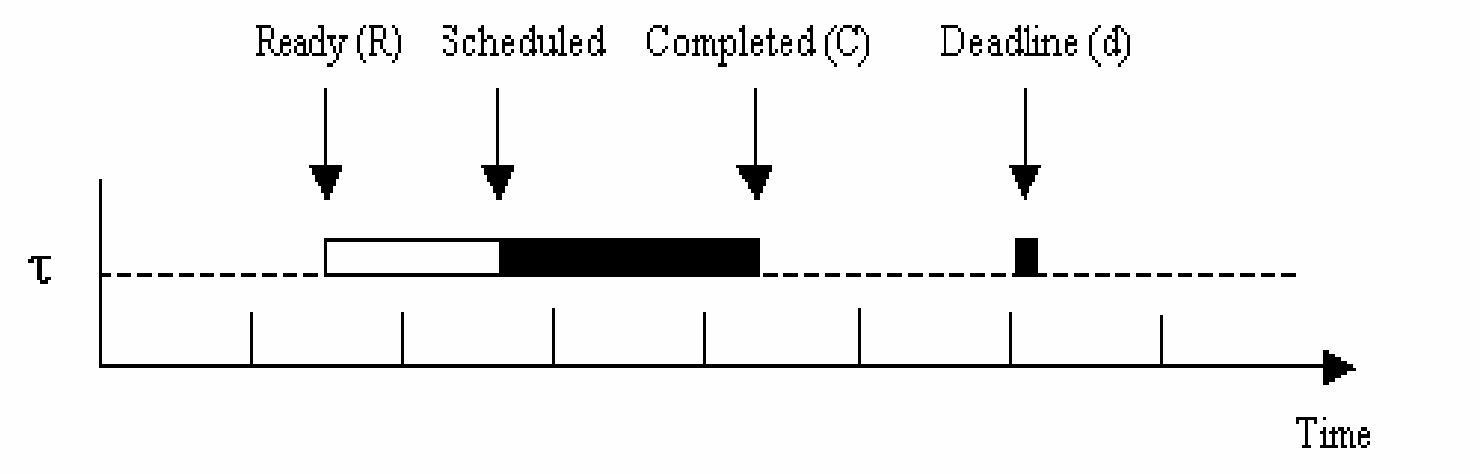
- Low cost and small size but able to manage at least 20 tasks
- Have minimum interrupt latency
- Deterministic execution time for all kernel services
- Provide at least the following services
  - Allow tasks to be dynamically created and deleted
  - Provide semaphore management services and interprocess communication (signals, events, ...)
  - Allow time delays and timeouts on kernel services (e.g delay of a task for a given time)
  - Management of special alarms
- Use of special file to accumulate data at a fast rate
- Preemptive scheduling based on priority with fast context switch
- Min interval during which interrupts are disabled

Time constraints

Definitions and Deadlines

Time Constrained Computation

- Computational activities in a real-time system generally have timing constraints



Time Constrained Computation

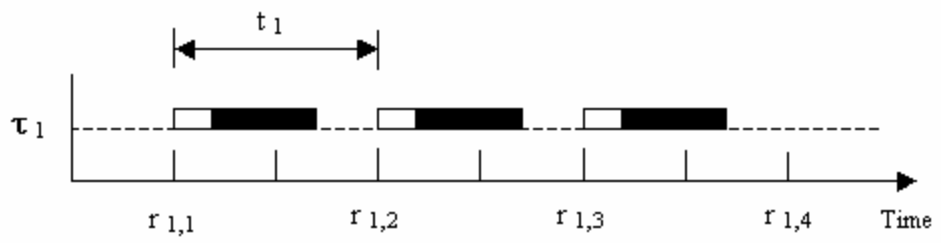
- The *Ready* event may be
  - Periodic
  - Aperiodic but predictable
  - Unpredictable
- The *Computation Time* may be
  - Fixed in duration
  - Variable
  - Unpredictable

Tasks and Jobs

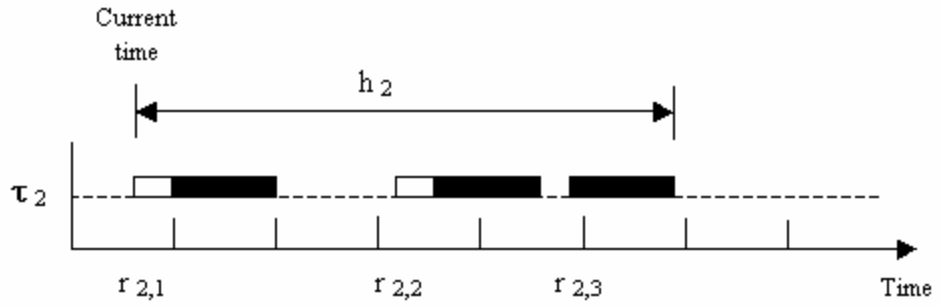
- In periodic computations there is a distinction between
  - Tasks: overall activity
  - Jobs: instantiations or individually scheduled computations of the task (thus a task is a stream of jobs)

Types of Computations

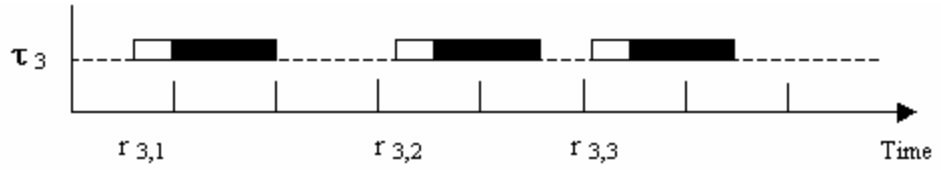
- Periodic Task



- Aperiodic, Predictable Task



- Aperiodic, Unpredictable Task

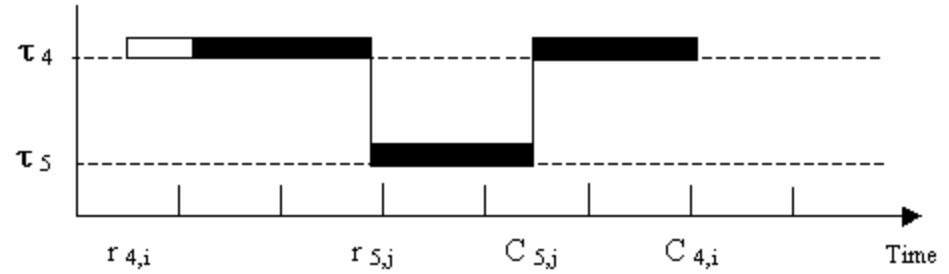


Preemptibility(1)

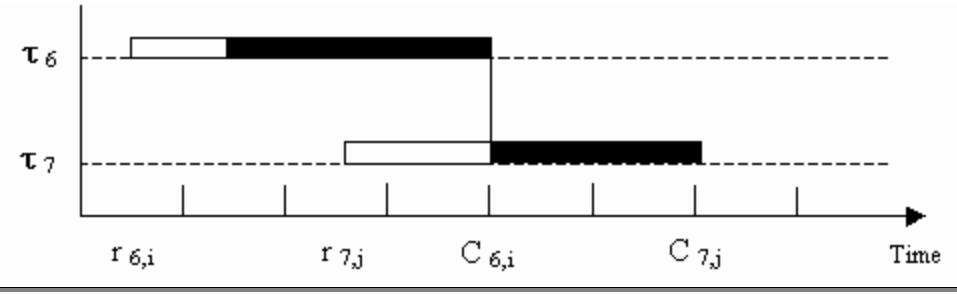
- A computation can be
  - Preemptible
  - Non-preemptible
  - Preemptible with one or more non-preemptible critical regions

Preemptibility(2)

- Preemptible task



- Non-premptible task



Nature of Deadlines

- *Hard* deadlines
  - The computation must be completed by the deadline time or a fatal error results
- *Soft* deadlines
  - The deadline may just be a recommendation or preference for completion of the computation

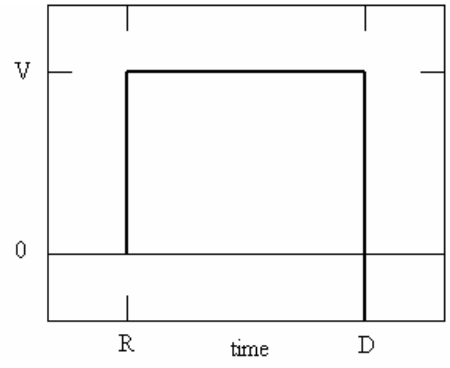
The Value Function

- A value function is a function of time which indicates the value that completion of the task would contribute to the overall value of the system
- Examples of value function…

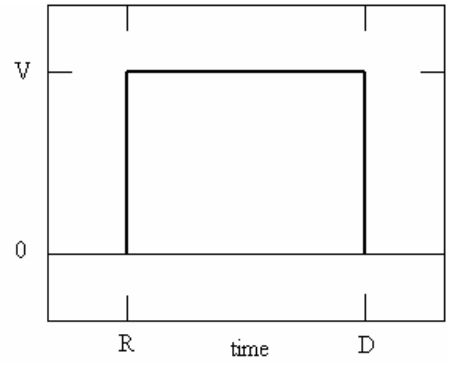
Non Real-Time VF

The Value Function(2)

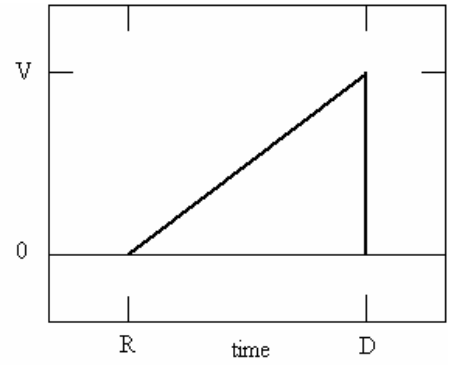
Catastrophic Deadline VF



Hard Deadline VF



Ramped Hard Deadline VF



Soft Deadline VF