Real-time Operating Systems

LISTA/UFSC

Prof. Dr. Antônio Augusto Fröhlich
Fauze Valério Polpeta
Lucas Francisco Wanner

http://www.lisha.ufsc.br/

March 2009
Real-Time Operating Systems

“A real-time application requires a program to respond to stimuli within some small upper limit of response time.”

A real-time operating system (RTOS) is designed to support real-time applications and therefore delivers its services under defined time constraints

(Foldoc)
Classes of Real-Time Systems

- **Hard real-time system**
  - Failure to meet deadlines is fatal
  - Validation by formal methods or extensive simulation
    - Flight control system

- **Soft real-time system**
  - Late completion of tasks is undesirable but not fatal
  - System performance degrades as more tasks miss deadlines
    - DVD player
What is Hard-TR? What is Soft-RT?
GPOS x RTOS

- General-purpose OS
  - Multiuser
    - time-sharing
    - access control, protection, system-call interface, etc
  - Applications
    - Independently run under the control of the OS

- Real-Time OS
  - Single user
    - determinism
    - relaxed access control and protection (if any)
  - Application
    - Tied together with the RTOS
RTOS Typical Features

- Scheduling
  - Deterministic algorithms
    - Usually some sort of priority
    - Predictable worst-case task *flyback* time
    - Concerns about queue manipulation

- Resource Management
  - Low-overhead
  - Aware of priority inversions

- Interrupt Handling
  - Guaranteed worst-case interrupt latency
Scheduling in RTOS

- Scheduling criteria
  - Priorities
  - Number of tasks
  - Resource requirements
  - Release time
  - Execution time
  - Deadlines
Taxonomy of Real-Time Scheduling

- Real-Time Scheduling
  - Soft
  - Hard
    - Dynamic
      - Preemptive
      - Nonpreemptive
    - Static
      - Preemptive
      - Nonpreemptive
Task Scheduling in RTOS

- Periodic tasks
  - Tasks with regular invocation times (period)
  - `wait_for_next_period()`
    - Sensor data processing
- Aperiodic tasks
  - Tasks with irregular invocation times
  - Handle random events or complement the execution of periodic tasks
  - Logging
Periodic Scheduling Algorithms

- Rate Monotonic (RM)
  - Preemptive static-priority scheduling algorithm in which tasks with shorter periods (deadline = period) are given higher priorities
    - Tasks with higher frequency will have higher priority
  - Optimal static-priority algorithm
    - No other fixed priority assignment rule can schedule a task set which cannot be scheduled by RM
  - Limitations
    - In general, all deadlines can be met if CPU utilization by RT tasks lays below 69.3%
Periodic Scheduling Algorithms

- Earliest Deadline First (EDF)
  - Preemptive dynamic-priority scheduling algorithm in which tasks closest to their deadlines are given higher priorities
  - Contrasts with RM, in which priorities do not change with time
- Limitations
  - Higher overhead than RM (dynamically compute priorities)
  - There is no way to guarantee which tasks will fail in a transient overload (with RM, low priority tasks always fail first)
Periodic Scheduling Algorithms

- Maximum Urgency First (MUF)
  - Mixed-priority (static/dynamic) algorithm in which each task is given an “urgency” defined by two static priorities plus a dynamic priority
    - Tasks with the highest urgency are scheduled first
  - Limitations
    - More difficult to implement
    - Requires a more clever task priority assignment
Aperiodic Scheduling Algorithms

- Pooling
  - The system periodically checks for aperiodic events, thus scheduling associated tasks

- Event-driven
  - Aperiodic events are handled as they occur

- Aperiodic server
  - Ticket-based algorithms
    - Server creates tickets according to a given policy
    - Aperiodic event handling consumes tickets
Aperiodic Scheduling Servers

- **Deferrable Server (DS)**
  - Tickets are replenished at regular intervals, independently of usage

- **Sporadic Server (SS)**
  - It preserves its server execution time at its high priority level until an aperiodic request occurs
  - It replenishes its server execution time to full capacity
  - Its aperiodic response time is comparable to that of the deferrable server but has a larger server size
Resource Management in RTOS

- **Memory**
  - Simple memory management schemes
    - Lower overhead, higher determinism
    - Most embedded processors do not feature a MMU

- **Storage devices**
  - Simple access protocols
  - Priorities inherited from tasks

- **Resources sharing**
  - May lead to priority inversions
  - Specific allocation/control algorithms
Priority Inversion

Task A gets a message in its queue and unblocks; RTOS switches to Task A.

Task B gets a message in its queue and unblocks; RTOS switches to Task B.

Task A tries to take the semaphore that Task C already has taken.

Task C takes a semaphore that it shares with Task A.

Task B goes on running and running and running, never giving Task C a chance to release the semaphore. Task A is blocked.

The task the microprocessor is executing
Interrupt Service Routines in a RTOS

- ISRs in RTOS are by their own sources of non-determinism for the system as a whole
  - Hardware interrupts are asynchronous events

- ISRs should care not to add on the matter
  - An ISR should not call blocking RTOS functions
  - An ISR can signal a context switch but should get itself involved in such an event
Interrupt Service Routines in a RTOS

- How ISRs should work

![Diagram showing interrupt service routines in a RTOS]

- ISR
- RTOS
- TaskHigh
- TaskLow

Send message to mailbox.

Time
Interrupt Service Routines in a RTOS

- What would really happen

- RTOS is unaware of ISRs, it switches to a high priority task and the ISR is delayed!
Interrupt Service Routines in a RTOS

- How ISRs do work (Plan A)

- RTOS know about ISRs, hardware interrupts
Interrupt Service Routines in a RTOS

- How ISRs do work (Plan B)

- The ISR suspend the scheduler
Nested Interrupts

- Higher priority ISR interrupts low-priority
  - When the higher priority ISR finishes, it must return to the low-priority ISR and not to a ready task