Real-Time Programming

Lecture 7

Farhang Nemati

Spring 2016

Repetition

- Message Passing
  - Synchronization Model, Naming of Source/Destination, The Structure of Message
- Message Queues
- Client-Server Communication
- Pipes
- Sockets
- Signals

Basic Facilities Provided by a RTOS

- Timing Facilities
- Task Management
- Memory Management
- Error Handling
- I/O Services
- Interrupt Handling
- Task Synchronization and Communication
- Scheduling

Basic Services Provided by a RTOS; Timing Facilities

- Clocks, Timers
- Time resolution
- Handling the interrupt issued by system tick:
  - Store the information and state of the executing task
  - Update (virtual) timers
  - Wake up periodic tasks if any and if their period has passed
  - Call scheduler, i.e., sort the tasks in the ready queue
  - Load the information and state of the task at the top of ready queue and assign it to the processor

Basic Services Provided by a RTOS; Task Management

- Creating tasks
- Deleting Tasks
- API to change task attributes

Basic Services Provided by a RTOS; Memory Management

- Dynamic memory management
- No paging and virtual memory for real-time tasks
- Memory protection is often not provided; because of performance and determinism

Basic Services Provided by a RTOS; Error handling

- Deadlocks, missing deadlines, memory management errors, etc.
  - Error code of systems calls, e.g., error codes in errno.h (POSIX)
  - Exception handling
  - Fault tolerance

Basic Services Provided by a RTOS; I/O Services

- Files, devices, derivers
- Create/Remove
- Open/Close
- Read/Write

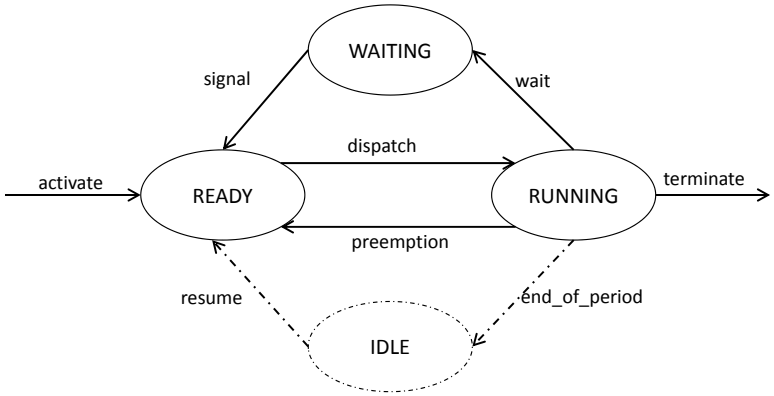
Basic Services Provided by a RTOS; Interrupt Handling

- An interrupt is usually used to communicate with hardware
- An interrupt is associated with an Interrupt Service Routine (ISR); the ISR is executed when the interrupt is arrived.
- In non real-time OS handling an interrupt is usually immediate to response to a device fast
  - In a RTOS this might interfere with hard real-time tasks
  - A task could be responsible for handling all interrupts

Basic Services Provided by a RTOS; Task Synchronization and Communication

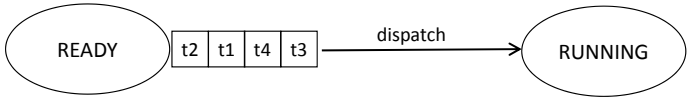
- Semaphores, mutexes
- Condition variables
- Monitors
- Message queues
- Signals

Real-Time Scheduling



Real-Time Scheduling

- How the tasks in the ready queue are ordered for getting the processor, i.e., transferring to Running
- The Scheduler provided by RTOS sorts the queue according to a scheduling algorithm



Real-Time Scheduling

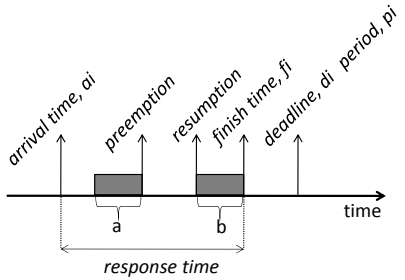
- Sort tasks in the ready queue based on:
  - Arrival time (FIFO)
  - Execution time
  - Priority
  - Deadline

Real-Time Scheduling

- Task Model
- Schedulability Analysis of a task set
- Scheduling Algorithms

Task Model

- Task parameters



execution time = a + b

Worst Case Execution Time (e_i) = max(execution time)

Task Model

- A set of tasks
- Each task with specified parameters
  - Depends on whether the tasks are periodic, aperiodic, or sporadic

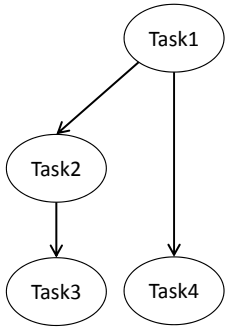
$$T = \{\tau_1, \tau_2, ...\tau_n\},$$
$$\tau_i(e_i, \rho_i, d_i, ...)$$

Task Model

- Resource requirement constraints
  - Mutual exclusive resources
  - Resource sharing protocols are needed

Task Model

- Precedence constraints



Task Model; Aperiodic Tasks

- Each task is specified by 3 parameters:
  - Arrival time, a_i
  - Execution time, e_i
  - Deadline, d_i

$$\tau_i(a_i, e_i, d_i)$$

Task Model; Periodic Tasks

- Each task is specified following parameters:
  - Execution time, e_i
  - Deadline, d_i
  - Period, p_i
  - Utilization factor ($u_i = e_i / p_i$)

$$\tau_i(e_i, d_i, p_i)$$

Task Model;
Sporadic Tasks

- Each task is specified by following parameters:
  - Execution time, e_i
  - Deadline, d_i
  - Minimum inter-arrival time, $e\min_i$

$$\tau_i(e_i, d_i, e\min_i)$$

Schedulability Analysis

- Schedulability: Given a task set and model, schedulability is to guarantee that all tasks will meet their deadline
- Feasible Schedule: Is a schedule that if the tasks are scheduled according to it, all tasks will meet their deadlines
  - There could be more that one feasible schedules
- Schedulability Analysis: Given a task set and model, to formally check whether there is any feasible schedule.

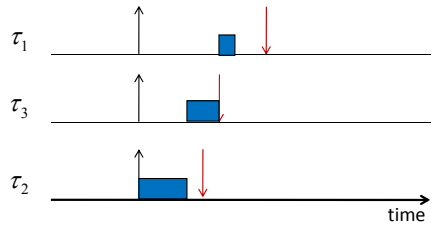
Schedulability Analysis

- Example, find a feasible schedule for following task set
  - with aperiodic model: $\tau_i(a_i, e_i, d_i)$
  - Assume tasks are independent (they don't share any resource other than processor)

$\tau_1(0,1,8)$
$\tau_2(0,3,4)$
$\tau_3(0,2,5)$

Schedulability Analysis

- A feasible schedule: $\tau_2(0,3,4)$ $\tau_3(0,2,5)$ $\tau_1(0,1,8)$



- There could be more feasible schedules

Categories of Scheduling Algorithms

- Preemptive vs. Non-preemptive
- Static vs. Dynamic
- Offline vs. Online
- Optimal vs. Heuristic
- Time driven vs. Event driven

Preemptive vs. Non-preemptive

- Preemptive Scheduling Algorithms: A running task can be preempted by higher priority tasks at any time during its execution
- Non-preemptive Scheduling Algorithms: When a task is started it will run to its end without being preempted by other tasks. Any scheduling decision can only be done once the task is finished

Static vs. Dynamic

- Static Scheduling Algorithms: A static algorithm is based on parameters of tasks (e.g., priority, etc.) that are assigned to the tasks before their execution
- Dynamic Scheduling Algorithms: A dynamic algorithm is based on parameters of tasks that may change during run time, e.g., the priority of a task could be decided based on the current state ready queue

Offline vs. Online

- Offline Scheduling Algorithms: An algorithm is offline if all scheduling decisions are taken offline, i.e., before the system starts running
- Online Scheduling Algorithms: An algorithm is considered online if the scheduling decisions are taken during run-time based on task parameters

Optimal vs. Heuristic

- Optimal Scheduling Algorithms: An optimal algorithm is able to find a feasible schedule if there exists one, i.e., if an algorithm is optimal and it cannot find a feasible schedule no other algorithm can find a feasible schedule either
- Heuristic Scheduling Algorithms: A heuristic algorithm is guided by a heuristic function. Its goal is find a solution optimal or close to optimal, however it does not guarantee finding an optimal solution

Time driven vs. Event driven

- Time driven Scheduling Algorithms: Scheduling decisions are triggered based on system time, i.e., arrival times of tasks are known beforehand.
- Event driven Scheduling Algorithms: Scheduling decisions are triggered based on arrival of events, i.e., arrival of tasks are not known in advance. Schedulability cannot be guaranteed since the number of arrivals of tasks is not deterministic.