Real-Time Programming

Farhang Nemati

Spring 2016

Repetition

- Embedded Systems
- Real-Time Systems
  - Hard Real-Time System
  - Soft Real-Time Systems
- Real-Time Task
  - Hard Task
  - Soft Task
  - Firm Task

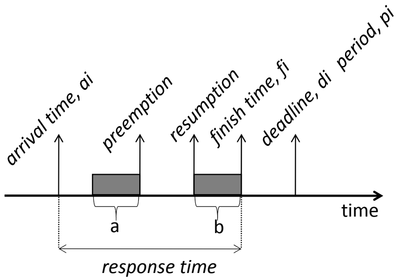
Properties of Real-Time Systems

- Complexity
- Reliability
- Concurrency
- Interactive with physical world
- Schedulability
- Fault Tolerant
- Predictability

Real-Time Systems

- Timing facilities of a RTOS
  - Clock
  - Delaying the execution of a task
- Task Communication
  - External communication
  - Synchronization
  - Data communication
- Extract Timing Requirements
  - Application requirements
  - Physical rules

Task parameters



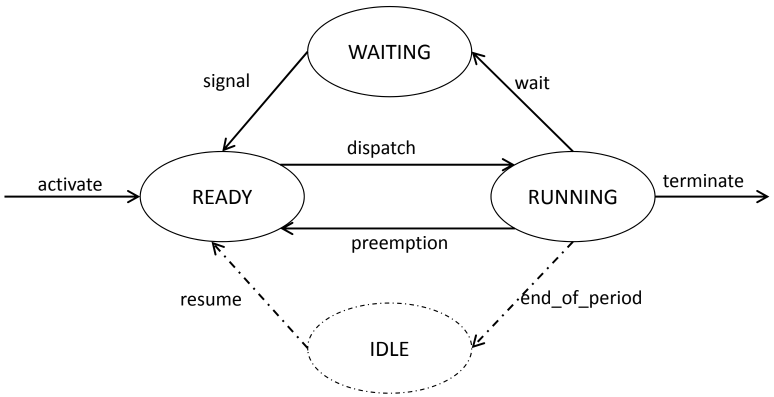
execution time = a + b

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

Real-Time Tasks

- Preemptive, non-preemptive
- Periodic
- Aperiodic
- Sporadic

Task States in RTOS



Task Synchronization and Communication

- Shared Variable Based
- Message Passing Based

Shared Variable Based Synchronization and Communication

- Atomic (Indivisible) Operation
- Critical Sections
- Mutual Exclusion
- Busy Waiting
- Semaphores
- Mutexes
- Condition Variables
- Monitors
- Barriers

Message Passing Based Synchronization and Communication

- Synchronization Model
  - Asynchronous
  - Synchronous
  - Remote Procedure Call
- Naming of Source/Destination
  - Direct Naming
  - Indirect Naming
  - Symmetry
- The Structure of Message

Task Synchronization and Communication

- Message Queues
- Client-Server Communication
- Pipes
- Sockets
- Events
- Signals

POSIX Standard

- POSIX = Portable Operating System Interface (for Unix)s
- A family of related standards
  - Provides standard Application Programming Interfaces (APIs) and command line shells and utilities for an operating system
  - Specified by IEEE
  - Facilitates developing applications portable to any operating system compatible with POSIX

Basic Facilities Provided by a RTOS

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

Real-Time Scheduling

- How the tasks in the ready queue are ordered
- The Scheduler provided by RTOS sorts the queue according to a scheduling algorithm
- A scheduling algorithm sorts tasks; decides which task should run next

Real-Time Scheduling

- Task Model
  - Task parameters
  - Resource requirement constraints
  - Precedence constraints
- Schedulability Analysis
  - Schedulability
  - Feasible Schedule
  - Schedulability test
- Scheduling Algorithms

Categories of Scheduling Algorithms

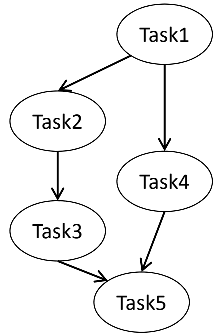
- Preemptive vs. Non-preemptive
- Static vs. Dynamic
- Offline vs. Online
- Optimal vs. Heuristic
- Time driven vs. Event driven

Aperiodic Scheduling Algorithms

- Same arrival times
  - Earliest Due Date (EDD) – Jackson's Algorithm
- Arbitrary arrival times
  - Earliest Deadline First (EDF) - Horn's algorithm
  - Least Slack Time First (LST)
  - Non-Preemptive:
    - Bratley's Algorithm
    - The Spring Algorithm (A heuristic algorithm)

Scheduling with Precedence Constraints

- Directed Acyclic Graph (DAG)



Scheduling Algorithms with Precedence Constraints

- Same arrival times
  - Latest Deadline First (LDF)
- Arbitrary Arrival times
  - EDF (not optimal)
  - Bratley's Algorithm
  - Spring Algorithm
  - EDF* (Transform arrival times and deadlines)

Periodic Scheduling Algorithms

- Timeline Algorithm (Cyclic Executive)
- EDF
- Rate Monotonic Scheduling Algorithm (RMS)
  - Critical instant of a task
  - Critical instant of a task set
  - Schedulability analysis
    - Sufficient but not necessary: $U_{\text{lub}} = n \times (2^{1/n} - 1)$
    - Response time analysis:

$$R^{(k+1)}{}_i = e_i + \sum_{\tau_j \in H_i} \left\lceil \frac{R^k{}_i}{p_j} \right\rceil e_j \qquad R^{(0)}{}_i = e_i + \sum_{\tau_j \in H_i} e_j$$

Jitter

$$jitter_i = delay_{\max} - delay_{\min}$$

where

$delay_{\max}$ = maximum delay of task τ_i from its period start
$delay_{\min}$ = minimum delay of task τ_i from its period start

Resource Sharing

- Priority Inversion
- Deadlock
- Chained blocking (Multiple blockings)
- Resource Access Protocols:
  - Non-Preemptive Protocol (NPP)
  - Priority Inheritance Protocol (PIP)
  - Highest Locker Priority Protocol (HLP). Also known as Immediate Priority Ceiling Protocol (IPC)
  - Priority Ceiling Protocol (PCP)

Petri Net

- A graphical and formal (mathematical) method for modeling and analyzing systems
- Modeling systems with concurrent and asynchronous processing
- Model and analysis in design phase

Petri Net

- Place
- Transition
- Arc
- Token
- Input-Places, Output-Places, Generator (Source), Stop (Sink)
- Enabled Transition, Firing a Transition
- Firing Sequence
- Weighted Arcs
- Marking
- Reachability Graph

Properties of Petri Nets

- Reachability
- Liveness
- Boundedness
- Fairness

Advanced Petri Nets

- Timed Petri nets
  - P-timed, T-timed
  - Stationary behavior
  - Maximum speed
  - Firing frequency
- Colored Petri Nets
  - Different values for tokens
  - Transitions sensitive to token colors (values)
  - Arcs associated with functions

Criteria to Pass the Course

- Approved written exam
  - 4-6 questions, 40 points
  - Required: 20 points for grade 3, 30 for grade 4, and 35 for grade 5
- Approved labs
  - Required: All 4 labs are handed in and demonstrated.

Example Exam

1. (10 points)

Briefly explain the following concepts:

a) Task
b) Semaphore
c) Priority Inversion
d) Message queue
e) Periodic, Aperiodic, Sporadic task
f) Deadlock
g) Monitor
h) Hard Real-Time Systems
i) Critical section
j) POSIX Standard

Example Exam

2. (12 points)

We want to implement a system which contains 3 periodic tasks; t1, t2, and t3. t1 and t2 run with period 10ms and t3 with period 8ms. Each task performs some work and send some data (an integer number) on a network bus. Sending on the bus take 1ms and only one task at a time can have access to the bus (mutually exclusive access).

a) Design the system using a P-timed Petri net
b) Show how the system can be implemented in a safe way

You can use the following function calls:

Example Exam

void work(int i); // performs the work for task i
void sendOnBus(int i); // sends data on the bus for task i
int nanosleep(const struct timespec* t1, struct timespec* t2); // the task for a given time in t1
int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);// creates a task
int pthread_mutex_init(pthread_mutex_t *restrict *mutex*, const pthread_mutexattr_t *restrict *attr*);
// initializes a mutex
int pthread_mutex_lock(pthread_mutex_t *mutex); // locks a mutex
int pthread_mutex_unlock(pthread_mutex_t *mutex); // unlocks a mutex

Example Exam

3. (6 points)

Show how the system in question 2 can be implemented

a) Using a monitor
b) Using an extra task, t4, that performs all the sending on the bus. I.e., the tasks t1, t2, and t3 put their data into a queue and t4 sends the data (an integer number) on the bus.

You may use the following function calls:

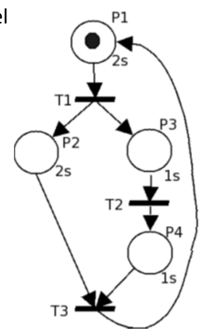
mqd_t mq_open(const char *name, int oflag, mode_t mode, struct mq_attr *attr);
//creates/opens a message queue
//O_RDONLY open the queue to receive messages only, O_WRONLY open the queue to
//send messages only, and O_RDWR open the queue to both send and receive messages.
int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len, unsigned int msg_prio);
//sends a message to the given queue
ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len, unsigned int *msg_prio);
//reads a message from the given queue

Example Exam

4. (6 points)

Assume that a system is designed with the following Petri net model

a) Draw the reachability graph of the Petri net
b) Assuming the model executes with maximum speed what is the
    1) Period of the Petri net
    2) The firing frequency of the transitions



Example Exam

5. (6 points)

a) Given a periodic preemptive task set, how the following scheduling algorithm would schedule the task set?
   1) Earliest Deadline First (EDF)
   2) Rate Monotonic Scheduling (RMS)

b) What is a Resource Access Protocol? How the following resource access protocols work?
   1) Priority Inheritance Protocol (PIP)
   2) Highest Locker Priority Protocol (HLP)

The End!

- Labs
  - Away: May 8th – 17^{th}, June 2 weeks
- The optional lab
- Please do the course evaluation!