Real-Time Software

Programming Real-Time Abstractions

René Rydhof Hansen

15 October 2010

Today's Goals

- Programming real-time abstractions
  - Cyclic executive
  - Priority-based scheduling under POSIX
  - Priority-based scheduling under Real-Time Java
  - Interrupt handling under Real-Time Java

Implementing Cyclic Executive "Scheduling"

- Reuqires a very simple run-time with
  - Regular timing interrupt
  - Table of procedure to call

Example

```
loop
   wait_for_interrupt;
   proc_a; proc_b; proc_c;
   wait_for_interrupt;
   proc_a; proc_b; proc_d;
   proc_e;
   wait_for_interrupt;
   proc_a; proc_b; proc_c;
   wait_for_interrupt;
   proc_a; proc_b; proc_d;
end loop;
```

Task	Period	Cost
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2

Priority-Based Scheduling

- Widely supported
  - Ada, Real-Time Java, POSIX
  - Most (commercial) RTOSs
- Usually preemptive
- Requires "reasonable" range of priorities
- Requires priority inheritance of some form

POSIX Priority Based Scheduling

Policies for Priority-Based Scheduling

- **FIFO:** A task runs until it completes or it is blocked
- **Round-Robin:** A task runs until it completes or is blocked or is pre-empted
- **Sporadic Server:** A task runs as a sporadic server
- **OTHER:** Implementation defined

Priorities

- Supports priority inheritance and ceiling protocols
- Dynamic priorities
- For FIFO and round-robin: at least 32 priorities must be supported

POSIX Priority Based Scheduling

Scheduler Granularity

- Scheduling policy can be set on a per thread basis
- Thread scheduling
  - System contention
    - Threads compete globally in a system
  - Process contention
    - Threads compete locally in a process
    - Scheduling unspecified relative to threads in other processes or threads with system contention

Other facilities in POSIX

- Priority inheritance to be associated with mutexes (ICPP)
- Message queues to be priority ordered
- Support for dynamically changing a thread's priority
- Threads determine if their attributes are inherited by child threads

Sporadic Server in POSIX

- A sporadic server assigns a limited amount of CPU capacity to handle events, has a replenishment period, a budget, and two priorities
- The server runs at a high priority when it has some budget left and a low one when its budget is exhausted
- When a server runs at the high priority, the amount of execution time it consumes is subtracted from its budget
- The amount of budget consumed is replenished at the time the server was activated plus the replenishment period
- When its budget reaches zero, the server's priority is set to the low value

Real-Time Profiles in POSIX

PSE51 Minimal real-time profile

- Threads, fixed priority scheduling, mutexes with priority inheritance, condition variable, semaphores, signals and simple I/O—analogous to Ravenscar

PSE52 Real-time control profile

- Multiprocessors, file system, message queues, tracing

PSE53 Dedicated real-time profile

- Multi-threaded processes, asynchronous I/O

PSE54 Multipurpose real-time systems profile

- Real-time and non real-time, memory management, networks etc.

Real-Time Java

- Scheduling at the level of objects: schedulable object
  - Extends notion of schedulability (for tasks, threads)
  - Schedulable object: any object implementing the `Schedulable` interface
- Scheduling parameters are represented by a class
- Enables online as well as static priority based scheduling
- Implementations are required to support at least 28 real-time priority levels
- Non real-time threads are given priority levels below the minimum real-time priority
- Like Ada and Real-Time POSIX, RTSJ supports a preemptive priority-based dispatching policy
- Unlike Ada and RT POSIX, RTSJ does not require a preempted thread to be placed at the head of the run queue associated with its priority level

`Schedulable` Interface

- Implemented by
  - `RealtimeThread`
  - `NoHeapRealtimeThread`
  - `AsyncEventHandler`
- Objects of these classes all have scheduling parameters

Schedulable Interface

```
public interface Schedulable
  extends java.lang.Runnable {
  ...
  public void addToFeasibility();
  public void removeFromFeasibility();

  public MemoryParameters getMemoryParameters();
  public void setMemoryParameters(MemoryParameters memory)

  public ReleaseParameters getReleaseParameters();
  public void setReleaseParameters(ReleaseParameters relea

  public SchedulingParameters getSchedulingParameters();
  public void setSchedulingParameters(
              SchedulingParameters scheduling);

  public Scheduler getScheduler();
  public void setScheduler(Scheduler scheduler);
```

RTSJ AsyncEventHandler

```
public class AsyncEventHandler
  extends java.lang.Object
  implements Schedulable
{
  public AsyncEventHandler(
              SchedulingParameters scheduling,
              ReleaseParameters release,
              MemoryParameters memory,
              MemoryArea area,
              boolean nonheap);
  ...
  public void handleAsyncEvent();
      // the program to be executed
  ...
  protected int getAndClearPendingFireCount();

 }
```

The SchedulingParameters Class

```
public abstract class SchedulingParameters {
  public SchedulingParameters();
}
public class PriorityParameters extends SchedulingParame
  public PriorityParameters(int priority);

  public int getPriority();
  public void setPriority(int priority) throws
                          IllegalArgumentException;
  ...
}
public class ImportanceParameters extends PriorityParame
  public ImportanceParameters(int priority, int importan
  public int getImportance();
  public void setImportance(int importance);
  ...
}
```

The Scheduler Class

- Mostly concerned with online (schedulability) tests

```
public abstract class Scheduler {
  protected Scheduler();

  protected abstract void addToFeasibility(
                          Schedulable schedulable);
  protected abstract void removeFromFeasibility(
                          Schedulable schedulable);

  public abstract boolean isFeasible();
  // checks the current set of schedulable objects

  public boolean changeIfFeasible(Schedulable schedulable,
                  ReleaseParameters release,
                  MemoryParameters memory);

  public static Scheduler getDefaultScheduler();
  public static void setDefaultScheduler(Scheduler scheduler);

  public abstract java.lang.String getPolicyName();
}
```

The PriorityScheduler Class

- Standard preemptive priority-based scheduling

```
class PriorityScheduler extends Scheduler
{
   public PriorityScheduler()
   protected void addToFeasibility(Schedulable s);
   ...
   public int getMaxPriority();
   public int getMinPriority();
   public int getNormPriority();

   public static PriorityScheduler instance();
   ...
}
```

Other Facilities in Real-Time Java

- Priority inheritance and ICCP (called priority ceiling emulation)
- Support for aperiodic threads in the form of processing groups; a group of aperiodic threads can be linked together and assigned characteristics which aid the feasibility analysis

Profile in Real-Time Java

Level 0 Similar to a cyclic executive

Level 1 FPS, periodic and sporadic events, mutual exclusion

Level 2 Asynchronous event handlers and/or `NoHeapRealtimethreads`, wait and notify

Interrupt Handling in Real-Time Java

- RTSJ views an interrupt as an asynchronous event
- The interrupt is equivalent to a call of the fire method
- The association between the interrupt and the event is achieved via the `bindTo` method in the `AsyncEvent` class
- The parameter is of string type, and this is used in an implementation-dependent manner—one approach might be to pass the address of the interrupt vector
- When the interrupt occurs, the appropriate handler's fire method is called
- Now, it is possible to associate the handler with a schedulable object and give it an appropriate priority and release parameters

Interrupt Handling

```
AsyncEvent Interrupt = new AsyncEvent();
AsyncEventHandler InterruptHandler = new
      BoundAsyncEventHandler(priParams,
                             releaseParams,
                             null, null, null);

Interrupt.addHandler(InterruptHandler);
Interrupt.bindTo("0177760");
```

Summary

Summary:

- Programming real-time abstractions
  - Cyclic executive
  - Priority-based scheduling under POSIX
  - Priority-based scheduling under Real-Time Java
  - Interrupt handling under Real-Time Java