

Real Time Systems



Slide 1

- **Lecturer:** Justin Pearson , email justin@docs.uu.se
- **Web page:** Not ready yet. It all appear soon, a link will be from <http://user.it.uu.se/~justin/Teaching/>

Course Text book



You have a choice of

Slide 2

- “Real-Time Systems” Jane W.S.Liu Prentice Hall
- “Real-Time Systems and Programming Languages”, Alan Burns and Andy Wellings. This book has more on Ada.
- There will also be an online set of note: “Fixed Priority Scheduling”, Tendell and Hansson.

Prerequisites



Slide 3

- Basic understanding of C.
- Basic understanding of Computer Architectures.
- Basic understanding of Operating System.

Aims of the Course



Slide 4

- To understand the basic requirements of real-time systems, and how to program such systems so that the requirements are met.
- To understand how these requirements have influenced the design of real-time programming languages and real-time operating systems.
- **For the 5 point version** To understand the implementation and analysis techniques which enable the requirements to be met.

Syllabus



Slide 5

1. Introduction: What are real-time systems?
2. Introduction to Scheduling
3. More on Scheduling, including rate monotonic and deadline monotonic
4. Blocking of resources, problems and solutions. Priority Inheritance, problems and solutions.
5. Fault Tolerance and Real-time Communication
6. **5 Point version:** Timed automata and the analysis of real time systems.

Introduction to real time systems



Slide 6

- What are real time systems?
- Main Characteristics of RT systems
- Desirable Properties of RT systems

Examples of Real Time systems

Slide 7

1. Medical Monitoring Systems
2. Car control systems, ABS systems, engine control system.
3. Fighter Plane Control Systems
4. Aircraft Simulators (or computer games which ever you find more interesting)
5. A chemical factory control system

What are the special design issues in each the examples?

Overall Structure of a RT System

Slide 8

- Hardware, CPU, I/O device etc.
 - a clock!
- A real-time OS (functions as a standard OS but with *predictable* behaviour and well defined functionality.
- A collection of RT tasks/processes that
 - share resources
 - Communicate/synchronise which each other and the environment.

Characteristics of Real Time Systems

A real time system is a system where:

- Slide 9**
- There is a continuous *interaction* with the environment (reactive system)
 - The environment changes in real time and imposes timing constraints on the systems
 - The system *simultaneously* controls and/or reacts to different aspects of the environment (concurrent systems)

Terminology

- Slide 10**
- **Reactive Systems:** Continuous interaction with the environment
 - **Time-sensitive systems:** Must react to the environment in time.
 - **Embedded systems:** Embedded in electronic and/or mechanical devices.
 - **Safety-critical systems:** A failure may cause the loss of lives.

Hard and Soft Systems



Slide 11

- Consider a control system for an aircraft. If the pilot is not giving new information about the environment quickly then the information is useless.
- It is no use being told what the air speed was 10 minutes ago.
- Also if a pilot gives a command (e.g. lower landing gear) he wants that done now, even a minute later could have some bad effect.

These systems are hard real time systems. (Solution don't use fly by wire)

Hard and Soft Systems



Slide 12

Consider a patient monitoring system, for example monitoring the

heart rate. It might be enough to know the heart rate within a second not the heart rate within the millisecond. You have to talk to Doctors to know this.

Programmers have to know their requirements.

Hard and Soft Systems

Consider a multimedia presentation system, this could be considered

Slide 13

a real-time system. But it doesn't really matter if all the deadlines are met on time as long as most of the time the deadlines are met.

Some systems have both hard and soft deadlines.

Soft real-time are hard to specify and require different approaches for hard real-time systems. In this course we shall concentrate on hard real time systems.

Hard and Soft Real time Systems

Slide 14

- Hard Real Time systems :- Events must happen at a certain time, the system should respond within some fixed time.
- Soft Real Time systems :- Exact Response time does is not critical.

A single systems may have hard and soft components. In reality many systems will have a cost function associated with missing each deadline.

Desirable Properties of a RT System

Slide 15

- Timeliness: the time output values are produced is important.
- Robustness: the system must not collapse when subject to a peak load.
- Predictability.
- Fault tolerance: Hardware and software failure must not crash the system.
- Maintainability: Modular structure to ease system modification.
- Testability: easy to test if the system can meet all deadlines.

Definition of a real time systems

Slide 16

Any system in which the time at which the output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to related to that same movement. The lag from input time to output must be sufficiently small for acceptable timeliness.

From the *Oxford Dictionary of Computing*

Deadlines



Slide 17

The specification for a real-time system often relate certain events in the environment with actions to be done.

For example if a car control systems recognises that the car is crashing then the driver airbag must be inflated within a certain time.

This bound is normally referred to as a *deadline*. Some deadlines are hard, some are soft.

We will assume that all deadlines are hard.

Real Time \neq Fast



Slide 18

Real time does not mean making every thing run fast.

Even if all your tasks run fast there might be problems due to interaction which would mean your results would come too late.

Many real time applications run on simple processors, tried and tested technology, which is better for mass-production a 50SEK controller or 2000SEK for some snappy pentium to control a washing machine?


Remember



Slide 19

- **Real Time does not mean fast**
- **Real Time means Just-In-Time (Predictable)**

Predictability: the most important requirement for a real-time system



Slide 20

- The system behaviour is known before it is put into operation.
- Difficult to achieve.

Predictability hard to achieve OS/Hardware

Slide 21

- Systems calls, difficult to know the worst case execution times
- Cache, hit ratio, pipelines, stalls.
- Interrupt handling may introduce unbounded delays
- Priority inversion (low-priority tasks blocking high priority task).
- Memory management
- Communication delays in a distributed environment.

Predictability hard to achieve Task level

Slide 22

- Difficult to calculate the worst case execution time for tasks (theoretically impossible \equiv halting problem). Practical methods:
 - Avoid dynamic data structures
 - Avoid recursion
 - Bounded loops
- Interaction between discrete and continuous components
 - Differential equations to solve
 - Control algorithms syntheses
- Complex synchronisation patterns between tasks.

Simple example

A temperature controller.

- Continuously read the current temperature (the sensor)
- adjust the heating element (actuator) depending on the temperature.

Slide 23

Implementing the rules:

- Whenever $T > 20C$ turn the heating element OFF
- Whenever $T < 20C$ turn the heating element ON

To specify this correctly you must know some physics and control theory. (If you end up working in real time you will have to know a lot more than how to write C code.)

The temperature Controller

Slide 24

- What makes a good controller?
- What makes a bad controller?
- A good controller keeps the temperature at around 20C all the time.
- A bad controller might oscillate to widely around 20C.

Tasks

Real time systems are concurrent. It is bad software engineering to

Slide 25

produce one big monolithic program which implements all the control.

Instead we split our problem into a number of tasks which then must be scheduled so that they can all complete their jobs at the right time.

Task = Process

Scheduling

Slide 26

It is then the job of the OS of the system or some supervisor program to execute the tasks at the required time.

Decided if a given set of deadlines can be met it referred to as *scheduling* theory.

Example System

Suppose we have car control system which has three software

processes:

- Slide 27**
- Speed measurement: $C = 10\text{ms}$, $T = 52\text{ms}$, $D=52\text{ms}$
 - ABS brake Control: $C = 10\text{ms}$, $T = 40\text{ms}$, $D=40\text{ms}$
 - Fuel Injection: $C= 10\text{ms}$, $T = 30\text{ms}$, $D=30\text{ms}$

Where C = Worst-case execution time; T = period, D = deadline.

Is it possible to schedule all these tasks to run on a single processor?

This is not so easy, RT OS

- Slide 28**
- Systems calls, difficult to know the worst case execution times
 - Cache, hit ratio, pipelines, stalls.
 - Interrupt handling may introduce unbounded delays
 - Priority inversion (low-priority tasks blocking high priority task).
 - Memory management
 - Communication delays in a distributed environment.