IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Course Plan
DIT161

Dept. of Computer Science and Engineering

DIT161, Real-Time Systems, 7.5 ECTS Credits

Advanced Level

1. Establishment

The Faculty Board at the IT-university established the course plan at 2006-11-17. This course plan is effective from autumn 2007.

Educational area: Technology/Sciences

2. Location

The course is a part of the Computer Science Master's programme and an elective course at Göteborg University.

3. Knowledge Requirements

The requirement for the course is to have successfully completed two years of an education aimed at a bachelor degree within Computer Science or equivalent.

4. Learning Outcomes

A real-time system is a computer system in which the correctness of the system depends on the time when results are generated. Real-time systems interact with a more or less time-critical environment. Examples of real-time systems are control systems for cars, aircraft and space vehicles, manufacturing system, financial transaction systems, computer games and multimedia applications. This course is intended to give basic knowledge about methods for the design and analysis of real-time systems.

After the course the students shall be able to:

- Construct concurrently executing software for real-time applications that interface to input/output units such as sensors and actuators.
- Describe the principles and mechanisms used for designing real-time kernels and run-time systems.
- Describe the mechanisms used for time-critical scheduling of tasks.
- Apply the basic analysis methods used for verifying the temporal correctness of a set of executing tasks.

5. Content

In the design of real-time systems it is practical to implement the application software as multiple concurrently executing processes, there each process is responsible for a given task in the system. The concept of multiple processes provides for an intuitive way of decomposing a complex system into smaller parts that are simple to comprehend and implement.

This course uses Ada as the main programming language because of its powerful support for programming of concurrent processes. In particular, the course demonstrates how language constructs such as rendezvous and protected objects are used for implementing communication/synchronization between processes, resource management and mutual exclusion. Since other programming languages uses monitors

or semaphores to implement these functions, the course also contains a presentation of such techniques. In addition, the course demonstrates how to use low-level programming in Ada to handle interrupt-driven communication with input and output devices. To demonstrate the general principles in real-time programming, the course also gives examples of how these techniques are implemented in other programming languages, such as C and Java.

In order to execute a program containing multiple concurrent processes there is a real-time kernel (run-time system) that distributes the available capacity of the microprocessor among the processes. The course shows how a simple real-time kernel is organized.

The real-time kernel determines the order of execution for the processes by means of a scheduling algorithm. To that end, the course presents techniques based on cyclic time-table based scheduling as well as scheduling techniques using static or dynamic process priorities. In addition, protocols for the management of shared hardware and software resources are presented.

In real-time systems with strict timing constraints it is necessary to make a pre-run-time analysis of the system schedulability. The course presents three different analysis methods for systems that schedule processes using static or dynamic priorities: utilization-based analyse, response-time analysis, and processor-demand analysis. In conjunction with this, the course also gives an account on how to derive the maximum resource requirement (worst-case execution time) of a process.

The course is organized as a series of lectures and a set of exercise sessions where the programming techniques and theories presented at the lectures are put into practice. The course material is examined by means of a final written exam. In addition, there is a compulsory laboratory exercise in which the students should implement the software for a realistic real-time application. Apart from the programming of cooperating concurrent processes, the exercise also encompasses advanced resource management and low-level programming of input and output devices.

6. Literature

See separate literature list.

7. Examination

Written exam.

8. Marks

The course is graded with the following marks: Fail, Pass, Pass with Distinction. The course can also, at the students' request, be marked according to ECTS standards.

9. Evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire can be used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

10. Other

The course is held in English.