13. Active objects and parallel programming

Description

A computer may execute many activities in parallel, which means they take place at the same time. This is possible since a computer may contain a number of processing units called CPU's or cores. Each core may execute an activity. A core may also share its processing time between a number of activities in a way that leaves the impression that they execute in parallel – this is called *time-slicing*. Activities executed in parallel are often referred to as *parallel process*es and/or *concurrent processes*.

A system of several computers connected in a network is another example of a system where activities are executed in parallel. This is referred to as *distributed computing* and/or *distributed processes*.

The writing of programs that execute just one activity is called *sequential programming* and the activity generated by the program is called a sequential process..

In the real world there are many example of systems where several active entities/agents work together on a common taks. This may be a group of people and machines building a bridge, a public office handling cases, … Such systems are complex in the sense that the entities have to coordinate their work and exchange information.

The ability of a computer to execute activities in parallel has many advantages. It may reduce the time for computing a given task. By splitting a given task into subtask that can be executed in parallel, may speed up computation compared to execute them in some order. However, there is an overhead in splitting up a task since these tasks may have to coordinate their activities. This means that the execution time of the subtasks must be larger than the time to coordinate the tasks.

The splitting of a task into subtasks may be the job of the programmers, but for some languages, the compiler behind the scene may split parts of a sequential program into subtasks that may be executed in parallel.

Another reason to consider parallel programming is that the environment consists of a number of devices, like computers, servers, and I/O-devices that all execute activities, and these cannot be replaced by a sequential program.

We refer to the above situations/cases for splitting a task into subtasks by the following terms:

- *Hidden*: made by the compiler
- *Exploited*: made by the programmers.
- *Inherent*: the system and its environment consist of devices executing in parallel.

In this book, we do not consider the hidden-case.

In this chapter, we will show how to describe parallel activities in a programming language. Parallel programming is a difficult task. This is partly due to the complexity of such systems, which makes it difficult for a single programmer or even more programmers to understand what goes on. Another reason is that the state-of-art for programming languages supporting parallel activities is not very well developed.