DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Real time systems: Process Model and Cyclic Executive

- Real TIme System Definition
- The process model
- Cyclic Executive

Real-time Systems

- In any concurrent program, the exact order in which processes execute is not completely specified.
  - The interprocess communication and synchronization primitives are used to enforce a set of constraints that limit the possible interleavings in order to ensure that the result of a concurrent program is correct in all cases.

- Nevertheless, the program will still exhibit a significant amount of nondeterminism because its processes may interleave in different ways without violating any of those constraints.

- The concurrent program output will of course be the same in all cases, but its timings may still vary considerably from one execution to another.

- If one of the processes in a concurrent program has a tight deadline on its completion time, only some of the interleavings that are acceptable from the concurrent programming perspective will also be adequate from the real-time execution point of view.

- As a consequence, a real-time system must further restrict the nondeterminism found in a concurrent system because some interleavings that are acceptable with respect to the results of the computation may be unacceptable for what concerns timings.

Real time Operating Systems

- Most general-purpose scheduling algorithms emphasize aspects such as,
  - **Fairness** In a general-purpose system, it is important to grant to each process a fair share of the available processor time and not to systematically put any process at a disadvantage with respect to the others.
  - **Efficiency** The scheduling algorithm is invoked very often in an operating system, and applications perceive this as an overhead. For this reason, the complexity of most general-purpose scheduling algorithms is forced to be O(1).i.e. it must not depend on how many processes there are in the system.
  - **Throughput** Especially for batch systems, this is another important parameter to optimize because it represents the average number of jobs completed

- The main goal of a scheduling model is to ensure that a concurrent program does not only produce the expected output in all cases but is also correct with respect to **timings**. In order to do this, a scheduling model must comprise two main elements:
  - 1. A **scheduling algorithm**, consisting of a set of rules for ordering the use of system resources, in particular the processors.
  - 2. An analytical means of analyzing the system and predicting its **worst-case behavior** with respect to timings when that scheduling algorithm is applied.

- In a hard real-time scenario, the worst-case behavior is compared against the timing constraints the system must fulfill, to check whether it is acceptable or not.

The assumptions of the basic model

- it turns out that the analysis of an arbitrarily complex concurrent program, in order to predict its worst case timing behavior, is very difficult.

- It is therefore necessary to introduce a simplified process model that imposes some restrictions on the structure of real-time concurrent programs to be considered for analysis.

- The simplest model, also known as the basic process model, has the following characteristics:
  - 1. The concurrent program consists of a fixed number of processes, and that number is known in advance.
  - 2. Processes are periodic, with known periods. Moreover, process periods do not change with time. For this reason, processes can be seen as an infinite sequence of instances. Process instances becomes ready for execution at regular time intervals at the beginning of each period.
  - 3. Processes are completely independent of each other
  - 4. Timing constrains are expressed by means of deadlines. For a given process, a deadline represents the upper bound on the completion time of a process instance. All processes have hard deadlines, that is, they must obey their temporal constraints all the time, and the deadline of each process is equal to its period.
  - 5. All processes have a fixed worst-case execution time that can be computed offline.
  - 6. All system's overheads, for example, context switch times, are negligible.

Limitations of the basic model

- Process independence must be understood in a very broad sense. It means that there are no synchronization constraints among processes at all, so no process must even wait for another.
  - This rules out, for instance, mutual exclusion and synchronization semaphores

- The deadline of a process is not always related to its period, and is often shorter than it.

- Some processes are sporadic rather than periodic. In other words, they are executed “on demand” when an external event, for example an alarm, occurs.

- For some applications and hardware architectures, scheduling and context switch times may not be negligible.

- The behavior of some nondeterministic hardware components, for example, caches, must sometimes be taken into account, and this makes it difficult to determine a reasonably tight upper bound on the process execution time.

Terminology

Symbol	Meaning
τ_i	The i-th task
$\tau_{i,j}$	The j-th instance of the i-th task
T_i	The period of task τ_i
D_i	The relative deadline of task τ_i
C_i	The worst-case execution time of task τ_i
R_i	The worst-case response time of task τ_i
$r_{i,j}$	The release time of $\tau_{i,j}$
$f_{i,j}$	The response time of $\tau_{i,j}$
$d_{i,j}$	The absolute deadline of $\tau_{i,j}$

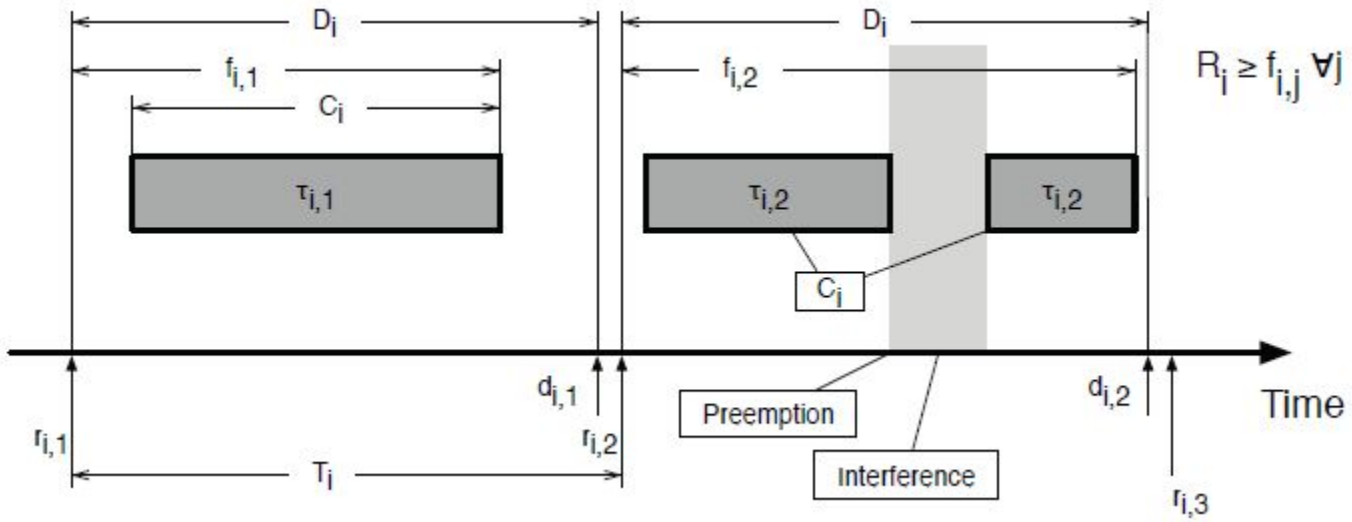
j-th volta che si esegue la task i

= Ci+ wait time

quando la task diventa ready

da ready a completa

Terminology

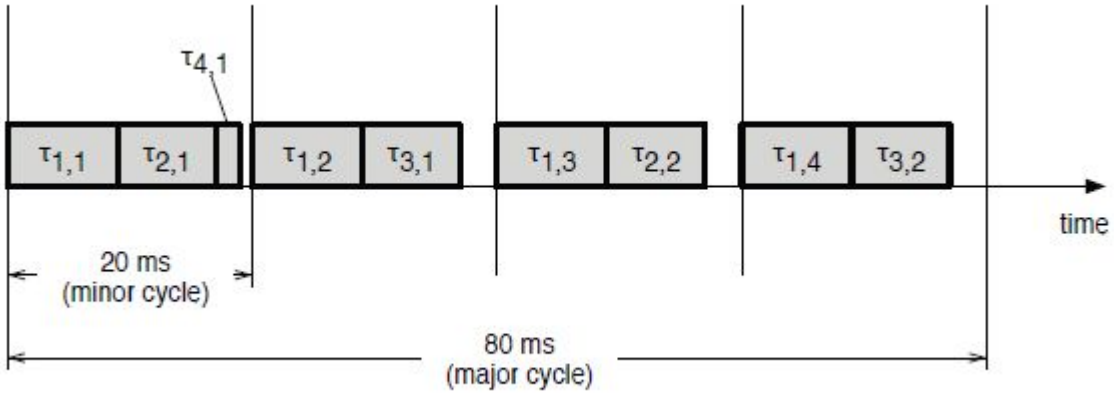


The Cyclic Executive

- The basic idea is to lay out ***offline*** a completely static schedule such that its repeated execution causes all tasks to run at their correct rate and finish within their deadline.
- For what concerns its implementation, the schedule can essentially be thought of as a table of procedure calls, where each call represents (part of) the code of a task.
- During execution, a very simple software component, the cyclic executive, loops through the table and invokes the procedures it contains in sequence.
- To keep the executive in sync with the real elapsed time, the table also contains synchronization points in which the cyclic executive aligns the execution with a time reference usually generated by a hardware component.
- The complete table is also known as the major cycle and is typically split into a number of slices called minor cycles, of equal and fixed duration.
- Minor cycle boundaries are also synchronization points: during execution, the cyclic executive switches from one minor cycle to the next after waiting for a periodic clock interrupt. As a consequence, the activation of the tasks at the beginning of each minor cycle is synchronized with the real elapsed time, whereas all the tasks belonging to the same minor cycle are simply activated

The Cyclic Executive - An example

Task τ_i	Period T_i (ms)	Execution time C_i (ms)
τ_1	20	9
τ_2	40	8
τ_3	40	8
τ_4	80	2




```
timer_setup(20);
while (1)
{
  wait_for_interrupt();
  task_1 ();
  task_2 ();
  task_4 ();
  wait_for_interrupt();
  task_1 ();
  task_3 ();
  wait_for_interrupt();
  task_1 ();
  task_2 ();
  wait_for_interrupt();
  task_1 ();
  task_3 ();
}
```

se avessi sia tau2 che tau3 con periodo 20ms il problema non potrebbe essere risolto con un solo processore

The Choice of Minor and Major Cycle time

- All task periods must be an integer multiple of the minor cycle period. Otherwise, it would be impossible to execute them at their proper rate.

- On the other hand, it is also desirable to keep the minor cycle length as large as possible. This is useful not only to reduce synchronization overheads but also to make it easier to accommodate tasks with a large execution time,

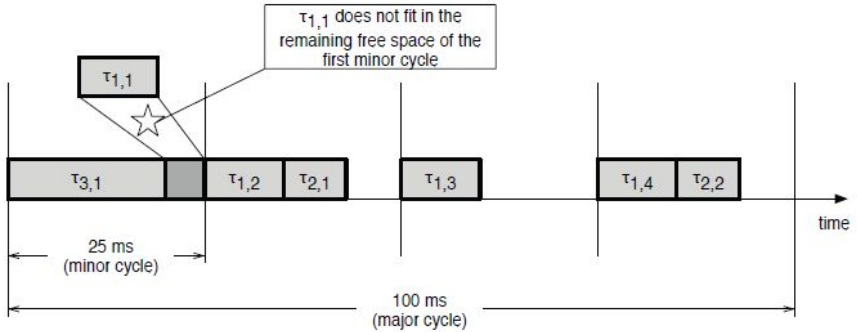
- To satisfy both constraints the minor cycle length is set to be equal to the **Greatest Common Divisor (GCD)** of the periods of the tasks to be scheduled.

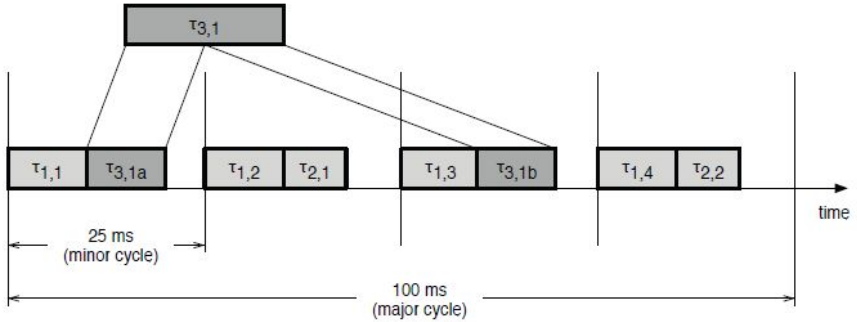
- The cyclic executive repeats the same schedule over and over at each major cycle. Therefore, the major cycle must be big enough to be an integer multiple of all task periods, but no larger than that to avoid making the scheduling table larger than necessary for no reason.

- A sensible choice is to let the major cycle length be the **Least Common Multiple (LCM)** of the task periods. Sometimes this is also called the hyperperiod of the task set.

Splitting tasks

Task τ_i	Period T_i (ms)	Execution time C_i (ms)
τ_1	25	10
τ_2	50	8
τ_3	100	20





con cyclic executive se NON ci sono split di task non ci dobbiamo preoccupare di race condition