Programmation concurrente et parallèle

riveill@unice.fr

http://www.i3s.unice.fr/~riveill

Au tableau

- Définition rapide d'un processus
  - Compteur ordinal
  - Pile
  - Tas
  - Vecteurs exceptions
  - Mémoire virtuelle
  - Couplage mémoire physique / mémoire virtuelle

- Définition rapide d'une thread
  - Plusieurs processus au sein d'un même processus

- Partage de données entre processus et entre threads (d'un même processus)

Why use concurrent programming?

- Natural application structure
  - The world is not sequential! Easier to program multiple independent and concurrent activities
- Increased application throughput and responsiveness
  - Not blocking the entire application due to blocking IO
- Performance from multiprocessor/multicore hardware
  - Parallel execution
- Distributes systems
  - Single application on multiple machine
  - Client/server or peer-to-peer systems

What is concurrency?

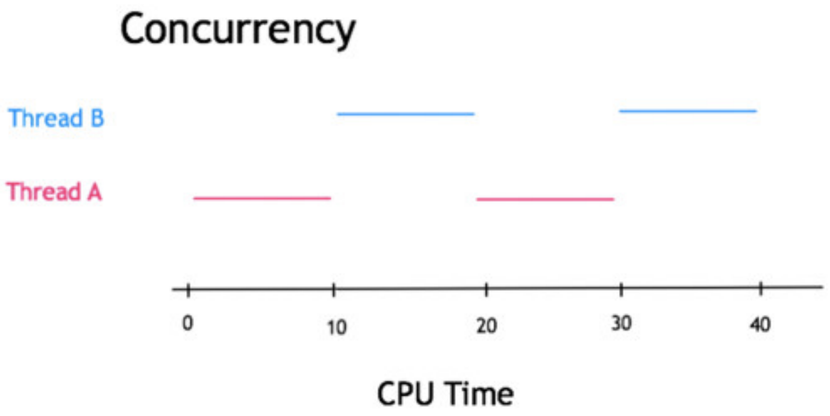
- What is a sequential program?
  - a single thread of control that executes one instruction and when it is finished execute the next logical instruction

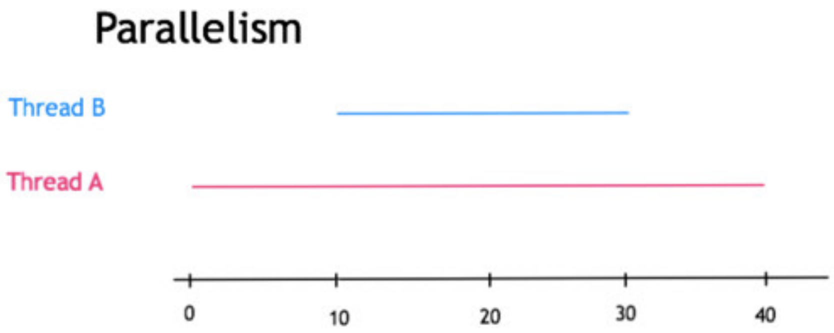
- What is a concurrent program?
  - a collection of autonomous sequential threads, executing (logically) in parallel

- The implementation (i.e. execution) of a collection of thread can be:
  - Multiprogramming
    - Threads multiplex their executions on a single processor
  - Multiprocessing
    - Threads multiples their executions on a multiprocessor or a multicore system
  - Distributed Processing
    - Processes multiplex their executions on a several machines

Concurrency and parallelism

- Concurrency is not (only) parallelism
- Interleaved concurrency
  - Logically simultaneous processing
  - Interleaved execution on a single processor
- Parallelism
  - Physically simulteous processing
  - Require a multiprocessors, a multicore system or a distributed system
- multi-core **processor**
  - Concurrency and parallelism





Synchronization

- All the interleavings of the threads are NOT acceptable correct programs
- All languages/systems provide synchronization mechanism to restrict the interleavings → Java, Python or Linux, Windows

- Synchronization serves two purposes:
  - Ensure safety for shared updates
    - Avoid race conditions
  - Coordinate actions of threads
    - Parallel computation
    - Event notification

Safety

- Multiple threads access shared resources simultaneously
- Safe only if:
  - All accesses have no effect on resource,
    - e.g. reading a variable

  or

  - All accesses idempotent
    - e.g. y = sinus(a) or a = 125

  or

  - Only one access at a time
    - **mutual exclusion**

- **SAFETY/sûreté : propriété la plus importante**
  - **Quelque chose de mauvais ne peut jamais arriver**

Safety

Si vous avez compris et mettez en œuvre ce principe simple

Exclusion des accès si modification

vous avez compris 90% du cours

Safety: the *"too much milk"* problem

You

```
arrive home
look in fridge
if (no milk) {
   leave for grocery
   arrive at grocery
   buy milk
   arrive home
}
drink milk
put milk in fridge
```

Is it correct ?

Safety: the *"too much milk"* problem

We try with 2 peoples

time	You	Your roommate
3:00 3:05 3:10 3:15	arrive home look in fridge if (no milk) { leave for grocery	
3:20		arrive home
3:25 3:30 3:35 3:40	arrive at grocery buy milk	arrive home look in fridge if (no milk) { leave for grocery
3:45 3:50 3:55	arrive home } drink milk	arrive at grocery buy milk
4:00 4:05 4:10 4:15	put milk in fridge	arrive home } drink milk put milk in fridge → **Oh no! too much milk!!!**

To keep in mind ...?
For the entire duration of the semester and if possible after

- The main challenge in designing concurrent programs is ensuring the correct sequencing of the interactions or communications between different computational executions, and coordinating access to resources that are shared among executions.
    - Potential problems include :
        - Race conditions / Compétition pour l'accès aux ressources partagées
        - Deadlocks / Interblocage
        - Resource starvation / Famine

- The wikipedia definitions of red words are excellent
    - Other articles to read about wikipedia
        - Concurrent_computing: presents the problem globally
        - Concurrency_control: presents means to solve problems but mainly in the context of databases

Samples

- *Concurrency is widespread but error prone.*

- Therac - 25 computerised radiation therapy machine
  - Concurrent programming errors contributed to accidents causing deaths and serious injuries.
  - http://en.wikipedia.org/wiki/Therac-25

- Mars Rover
  - Problems with interaction between concurrent tasks caused periodic software resets reducing availability for exploration.

Data races

- Problem with data races: non-determinism
  - Depends on interleaving of threads
- Usual question
  - ♦ *Is the system safe?*
  - ♦ *Would testing be sufficient to discover all errors*
- In "sequential programming"
  - Safe programming is easy, we use
    - Pre and post condition
    - Invariant
- With concurrent programming
  - All interleaving execution could be safe

→ we need a new approach to explore all the solution

→ We need a model in order to evaluate all possible execution

Models

- A model is a simplified representation of the real world.
- Engineers use models to gain confidence in the adequacy and validity of a proposed design.
  - focus on an aspect of interest - concurrency
  - model animation to visualise a behaviour
  - mechanical verification of properties (safety & progress)
- Models are described using state machines, known as Labelled Transition Systems **LTS**. These are described textually as finite state processes (**FSP**) and displayed and analysed by ***LTSA*** (Labelled Transition Systems Analysis tool).

Modeling problem

- It's not so easy

- It is necessary to model only what is necessary to prove
  - Is a model too big → prof is too complicated, and tool can't work

  - Is an important thinks was not included in the model → prof is incorrect

Our modelling tool

- **Based on model-checking or temporal logic results**

- **2 parts**
  - **finite state processes (FSP) - algebraic form**
    - to model processes as sequences of actions.
  - **labelled transition systems (LTS / LTSA) – graphical form**
    - to analyse, display and animate behavior.

  - FSP - algebraic form
  - LTS - graphical form
  - LTSA – analysing tools

Q&A

http://www.i3s.unice.fr/~riveill