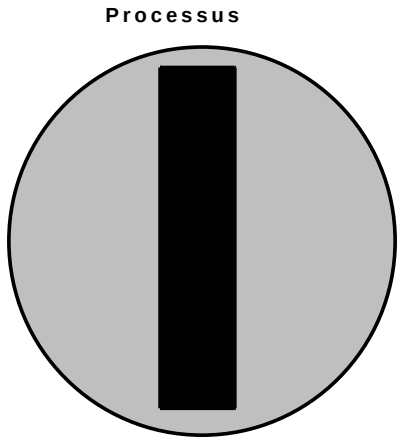
Système partie 1

Pastille 2 - les threads

N. de Rugy-Altherre - Vincent Colotte

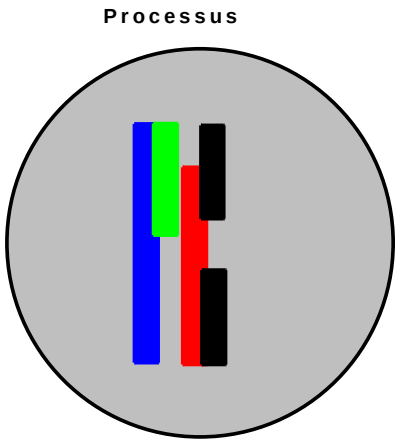
Les threads

Classiquement : un processus tourne à la fois



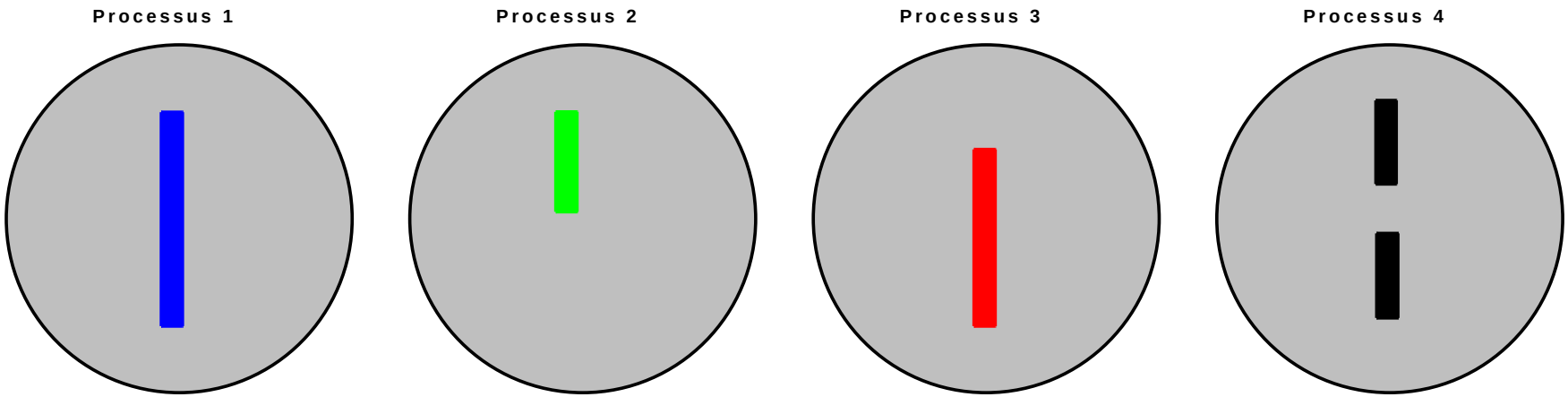
Les threads

On voudrait faire plusieurs tâches en même temps sur le même processus



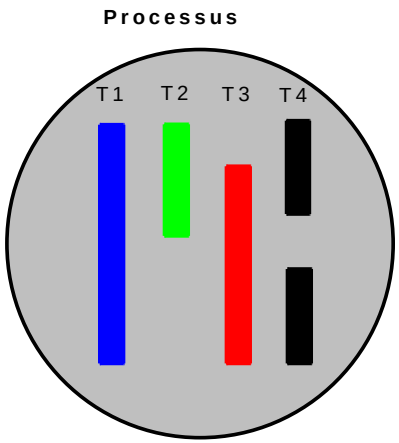
Les threads

On pourrait faire ces tâches sur plusieurs processus



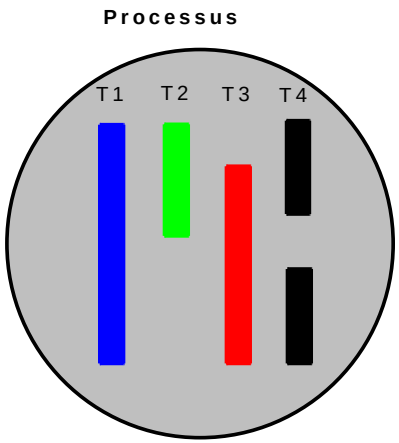
Les threads

Mais on peut aussi mettre ces tâches un seul processus, en même temps, sur plusieurs threads



Les threads

Mais on peut aussi mettre ces tâches un seul processus, en même temps, sur plusieurs threads



Thread :

- “Processus léger” : code, pile user.
- Données communes aux autres threads
- Communication plus aisée (mais synchronisation)

Les threads

Du point de vue du SE : 3 stratégies

Approche "user" Mapping N :1 $\Rightarrow$ considérés comme un même processus (ordonnancement invisible au système).

Les threads

Du point de vue du SE : 3 stratégies

Approche "user" Mapping N :1 $\Rightarrow$ considérés comme un même processus (ordonnancement invisible au système).

Approche "noyau" Mapping 1 :1 $\Rightarrow$ 1 thread = 1 “processus” (Unix, Linux, MacOS) (*)

Les threads

Du point de vue du SE : 3 stratégies

Approche "user" Mapping N :1 $\Rightarrow$ considérés comme un même processus (ordonnancement invisible au système).

Approche "noyau" Mapping 1 :1 $\Rightarrow$ 1 thread = 1 “processus” (Unix, Linux, MacOS) (*)

(*) peut donc tirer profit d'un multi-coeur ou multi-processeur.

Les threads

Du point de vue du SE : 3 stratégies

Approche "user" Mapping N :1 $\Rightarrow$ considérés comme un même processus (ordonnancement invisible au système).

Approche "noyau" Mapping 1 :1 $\Rightarrow$ 1 thread = 1 “processus” (Unix, Linux, MacOS) (*)

Approche "mixte" Mapping N :M $\Rightarrow$ M (<N) “processus” à ordonnancer (Windows, Java) (*)

(*) peut donc tirer profit d'un multi-coeur ou multi-processeur.