La concurrence

dans Smalltalk

Noury Bouraqadi
http://car.mines-douai.fr/noury

Option ISIC
Ecole des Mines de Douai

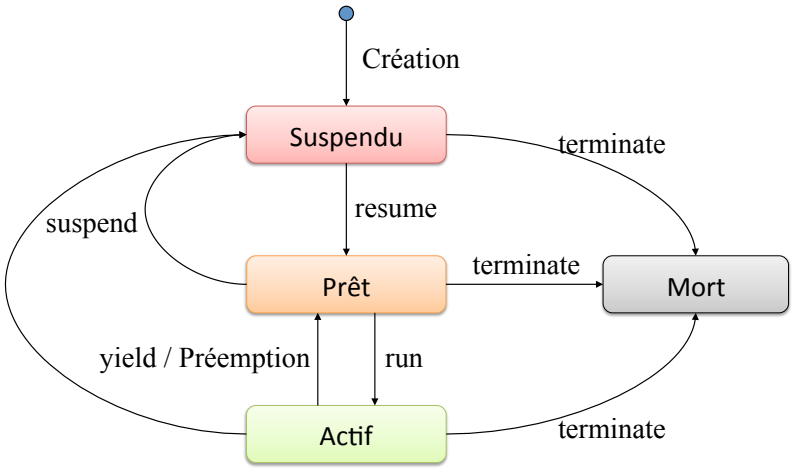
Programmation concurrente

- **Concurrence = multi-tâche ~ parallélisme**
- **Objectif : réaliser plusieurs activités "simultanément" au sein du même programme**
- **Applications :**
  - Simulation (représentation d'entités actives) ,
  - Réseau (connexions multiples),
  - Applications interactives,
  - programmes avec E/S

Threads = Processus légers

- **1 thread = 1 "fil" d'exécution**
  - A l'intérieur d'un thread l'exécution est séquentielle
- **Caractéristiques des thread :**
  - Exécution dans le même espace d'adressage
  - Exécution concurrente
  - Géré par la machine virtuelle
- **En Smalltak**
  - Utilisation du terme "processus" pour désigner les threads
  - threads = instances de la classe Process
  - Tout traitement se passe dans un thread
    - Gestion souris, sauvegarde, …

Cycle de vie d'un thread Smalltalk



Ordonnencement des thread Smalltalk

- **Nombre arbitraire de threads >= 1**
  - Mais toujours 1 seul actif = en cours d'exécution
  - Par défaut : thread qui boucle et ne fait rien
- **Chaque thread a une priorité**
  - thread actif = a la plus haute priorité parmi ceux prêts
  - Le plus ancien parmi plusieurs threads de priorités égales
- **Ordonnenceur = instance unique de ProcessorScheduler**
  - Variable globale **Processor**
  - Donne les différentes priorités valides
    - protocole "priority names"

Changement de thread actif

- **Préemption** (automatique)
  - Un thread de priorité supérieure à celle du thread actif est prêt, il devient actif.
  - L'ancien actif retourne à la liste des processus prêts en attente
- **Yield** (manuel = codé par le développeur)
  - Un thread peut permettre aux autres threads de même priorité de devenir actif
    - Il se place à la fin de la liste d'attente des processus prêts
  - Le bloc de traitements qu'il exécute doit contenir
    - **Processor yield**

Création d'un thread Smalltalk

- **Utilisation des blocs (cf. classe BlockClosure)**
  - newProcess
    - Crée un thread suspendu
    - Même priorité que le thread actif
  - fork
    - Crée un thread prêt à l'exécution
    - Même priorité que le thread actif
  - forkAt: priorité
    - priorité = nombre entre 10 et 80
    - Voir le protocole "priority names" de la classe ProcessorScheduler

Exemple de création de thread Smalltalk

- [100 timesRepeat: [
  Transcript cr; show: 'ping'].
] forkAt: Processor userBackgroundPriority.

- |monThread|
monThread := [50 timesRepeat: [
  Transcript cr; show: 'PONG']] newProcess.
monThread resume

Attentes (Delay)

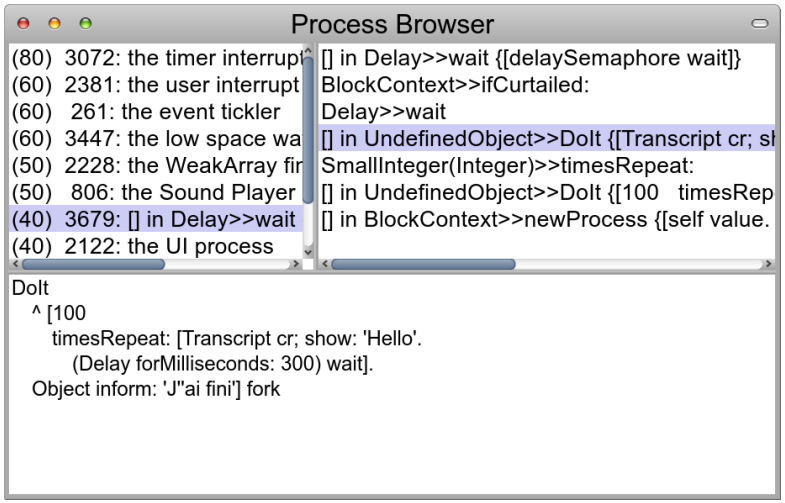
- **Objectif :**
  - Suspendre l'exécution d'un thread pendant une durée précise
    - Le thread est alors dans l'état "suspendu"
- **Besoins typiques :**
  - Traitement répété tous les x milliseconds
    - Exemple : Lire un capteur de température
  - Attendre un certain temps avant de réaliser une action
    - Exemple : Attendre 1 min avant de passer le feu du vert au rouge
- **Utilisation de la classe Delay**

```
|attente|
attente := 300 milliSeconds asDelay. "Création d'un délai"
attente wait. "Suspendre le processus en cours"
```


Exemple d'utilisation des attentes

```
[|attenteEntrePing|
attenteEntrePing := Delay forMilliseconds: 300.
10 timesRepeat: [
    Transcript cr; show: 'ping'.
    attenteEntrePing wait].
] forkAt: Processor userBackgroundPriority.
[|attenteEntrePONG|
attenteEntrePONG := Delay forMilliseconds: 100.
10 timesRepeat: [
    Transcript cr; show: 'PONG'.
    attenteEntrePONG wait].
] forkAt: Processor userBackgroundPriority.
```


Process Browser (Pharo)



Besoin de synchronisation

- **L'accès simultané à une même ressource par plusieurs processus légers peut entraîner des incohérences**

Thread A : cpt debiter: 1000	*Thread B : cpt debiter: 200*
Lecture solde := 3000	
Soustraction solde – 1000	***Lecture solde := 3000***
Ecriture solde := 2000	
	Soustraction solde – 200
	Ecriture solde := 2800

Synchronisation par exclusion

- **Marquer des portions de code critique comme étant**
  - mutuellement exclusives
  - auto-exclusives
- **Lorsque 2 threads essayent d'exécuter 2 portions de code mutuellement exclusives**
  - Un seul thread s'exécute
  - Le second attend que le premier ait fini la portion critique
- **La classe de verrous Mutex**
  - A utiliser dans les portions de code critiques
  - Une instance = verrou permet la synchronisation

Exemple de synchronisation avec un Mutex

```
|verrou compteur|
verrou := Mutex new.
compteur := 0.
[10 timesRepeat: [Processor yield.
        verrou critical: [compteur := compteur + 1.
                Transcript cr; show: compteur]
]] fork.
[10 timesRepeat: [Processor yield.
        verrou critical: [compteur := compteur - 1.
                Transcript cr; show: compteur]
] ] fork
```


Exemple de synchronisation [illegible] Mutex

Laisser la place aux autres theads de même priorité

```
|verrou compteur|
verrou := Mutex new.
compteur := 0.
[10 timesRepeat: [Processor yield.
        verrou critical: [compteur := compteur + 1.
                Transcript cr; show: compteur]
]] fork.
[10 timesRepeat: [Processor yield.
        verrou critical: [compteur := compteur - 1.
                Transcript cr; show: compteur]
] ] fork
```


Syncrhonisation par signaux

- **Idée : Un processus attend la réception d'un signal**
  - Attente + Signaux via des verrous

```
|verrou compteur|
compteur := 0.
verrou := Semaphore new.
[10 timesRepeat: [
   verrou wait.
   compteur := compteur + 1.
   Transcript cr; show: compteur.
   verrou signal
] ] fork.
```

```
[10 timesRepeat: [
   compteur := compteur * 10.
   Transcript cr; show: compteur.
   verrou signal
   verrou wait.
] ] fork.
```

Files de communications

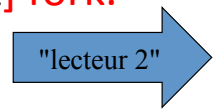
Noury Bouraqadi - option ISIC - Dépt. I. A.

- **Instances de la classe SharedQueue**
  - Canaux d'échange d'objets
    - Analogues aux sockets/streams ou pipes unix
  - Mais, synchronisés
    - 1 écrivain ou 1 lecteur à la fois
    - Les lecteurs sont bloqués quand la file est vide

Exemple de files de communications - 1

Noury Bouraqadi - option ISIC - Dépt. I. A.

```
|file|
file := SharedQueue new.
[[|nombre| "lecteur 1"
  nombre := file next.
  nombre ifNil: [Processor terminateActive].
  Transcript cr; show: nombre] repeat] fork.
```



Exemple de files de communications - 2

Noury Bouraqadi - option ISIC - Dépt. I. A.

```
[|compteur| "lecteur 2"
 compteur := 0.
 [file next ifNil: [
   Transcript cr; show: 'total = '; show: compteur.
   Processor terminateActive].
  compteur := compteur + 1] repeat] fork.
```

Exemple de files de communications - 3

Noury Bouraqadi - option ISIC - Dépt. I. A.

```
[|attenteEntreEcritures|
   attenteEntreEcritures := Delay forMilliseconds: 100.
   1 to: 10 do: [:entier|
    attenteEntreEcritures wait.
    file nextPut: entier].

   2 timesRepeat: [file nextPut: nil] ] fork.
```