Programmation de processus concurrents, en go

1 Généralités : go

En plus des *packages* de base comme `fmt`, `os` et `bufio` pour les entrées-sorties, `go` offre divers *packages* adaptés aux types de programmation à faire : `"net"` pour la programmation réseau, `"sync"` pour la programmation concurrente, etc Il faut donc importer les packages en fonction des besoins.

```
import (
    "net"  // for socket resources
    "os"   // for OS resources
    "fmt"  // general formatting
    "time" // tie handling resources/functions
    "sync" // mutex functions
)
```

2 Quelques fonctions pour illustrer

```
time.Sleep(time.Second) // make the time passed for 1 second
time.Sleep(time.Millisecond) // make the time passed for 1 millsecond
var smutex = &sync.Mutex{} // create a MUTEX variable
```

3 Expérimentations : processus concurrents, section critique, exclusion mutuelle

Vous avez à votre disposition en ligne sur la page du cours, plusieurs programmes. Ces programmes décrivent des expériences (avec plusieurs variantes) à travers lesquelles vous aller étudier et mettre en œuvre en `go` les mécanismes déjà étudiés en cours, notamment :
- l'exécution concurrente de plusieurs processus,
- le non-déterminisme des exécutions, et
- la mise en place des sections critiques (avec des exclusions mutuelles) quand c'est nécessaire (ie chaque fois qu'il y a une section critique imposée par l'accès à une ressource partagée).

Nous utilisons dans ces expériences, l'écoulement du temps, pour simuler le déroulement d'un processus (qui dans un projet, consommerait le temps pour effectuer des traitements).

Environnement de travail : Préparez un répertoire de travail, dans lequel vous allez copier les fichiers indiqués, puis procéderes à vos expérimentations.

Avant de commencer les expériences,
assurez vous d'avoir en mémoire (sinon regarder de nouveaux votre cours) :
- différence entre **exécution concurrente** et **exécution parallèle**
- *race condition*, section critique,
- non-déterminisme,
- interblocage.

Expérience 1 : exécution concurrente de plusieurs processus Copiez le programme suivant : `processus-goroutines_1.go`
et exécutez plusieurs fois le programme et observez son résultat.
linux> `go run processus-goroutines_1.go`

Copiez et exécutez (plusieurs fois) les programmes suivants (2 puis 3) : `processus-goroutines_{2,3}.go`
linux> `go run processus-goroutines_1.go`
Pourquoi on vous demande d'exécuter plusieurs fois le même programme avant de tirer des conclusions ?

Rédigez ce que vous avez compris, en un paragraphe concis, en utilisant les concepts appris.

Expérience 2 : concurrence entre plusieurs processus + accès à une ressource Copier et exécuter (plusieurs fois) les programmes : `sharedUpdates_{1,2,3}.go`

Rédigez ce que vous avez compris, en un paragraphe concis, en utilisant les concepts appris.

Expérience 3 : exécution concurrente de plusieurs processus + canal partagé Copier et exécuter (plusieurs fois) le programme : `sharing_1.go`

Rédigez ce que vous avez compris, en un paragraphe concis, en utilisant les concepts appris.

Méthode de travail A l'issue de ces expériences, vous devez avoir, une synthèse d'une page de ce que vous compris (ou pas). Cette synthèse doit contenir le *qui-quoi-quand-où*, et bien sûr un paragraphe d'introduction et un paragraphe de conclusion. N'hésitez pas à utiliser les ressources en ligne pour vérifier orthographe, conjugaison, etc (`larousse.fr`,`verb2verbe.com`, ...)

Ressources et environnement de travail

Page du cours `http://pagesperso.ls2n.fr/~attiogbe-c/mespages/programmation-applis-reparties.html`

— Environnement Linux (salles machine, sur le réseau univ-nantes)
— Langage Go et ses bibliothèques
— et autres liens conseillés :
`https://golang.org/pkg/net/` (Sockets en `go`)
`https://tour.golang.org/concurrency/1` (Introduction à la concurrente)
`https://gobyexample.com/mutexes`