Sync packet

MUTEX IN GO

В Golang рекомендуется решать проблемы синхронизации между горутинами при помощи каналов, специально придуманного для этого концепта. Но Каналы не могу решить всех проблем и могут встретиться сценарии, когда нужно будет обратиться к традиционным способам. Они как раз содержаться в пакете Sync.

GOLANG APPROACH

```
var x []int
go func() { x = make([]int, 10) }()
go func() { x = make([]int, 100000) }()
x[9999] = 1
```

Когда результат выполнения программы зависит от порядка исполнения

СОСТОЯНИЕ ГОНКИ

Примитив синхронизации работы процессов и потоков, в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующейся

СЕМАФОРЫ

Mutex или mutual exclusion lock это один из способов безопасной работы с разделяемым участком кода с множеством горутин. Использование Mutex это стандартный способ решить вопрос с проблемами конкурентности, таких как состояние гонки

MUTEX

```
func doSomething(){
    mu.Lock()
    http.Get() // какой-нибудь дорогой IO-вызов
    mu.Unlock()
}
```

Атомарные операции более примитвны чем другие механизмы синхронизации. Atomic также используются для реализации других техник синхронизации

ATOMIC

Это расширение mutex, которое добавляет два новых метода RLock и RUnlock. Это дает больше гибкости для операций чтения, теперь в этот момент программа не будет блокироваться

RWMUTEX

WaitGroup используется для координации в случае, когда программе приходится ждать окончания работы нескольких горутин.

WAITGROUP

Позволяет определить задачу для однократного выполнения за всё время работы программы. Содержит одну-единственную функцию Do, позволяющую передавать другую функцию для однократного применения

ONCE