Week 2

Concurrency

Trainers

Trầm Xuân Chuyển

Back-end Engineer

Nguyễn Ngọc Chiển

Back-end Engineer

Group chat

https://m.me/join/AbY3N1ooper5eALl

Agenda

- What is Concurrency ?
- How can Concurrency be achieved in Go ?
- References

What is concurrency ?

?

What is concurrency ?

Concurrency is the ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, without affecting the final outcome. (wikipedia)

How can Concurrency be achieved in Go ?

?

How can Concurrency be achieved in Go ?

- **Goroutines**
- **Channels**

Example

```
func hello() {
  fmt.Println("Hello Goroutine!")
}

func main() {
  go hello()
  fmt.Scanln()
}
```

Goroutine

What is Goroutine?

A **goroutine** is a lightweight thread managed by the Go runtime. It is just a function executing concurrently with other functions

Goroutine

Syntax

```
go function(...)
```

Channel

A **channel** is used to exchange data between goroutines

Channel

Type syntax

```
chan Type
```

Example

```
messages := make(chan string)
messages <- "hello" // send data
msg := <- messages  // receive data
```

Channel

```
n := make(chan int)      // unbuffered channel
s := make(chan int, 10) // buffered channel

func receive(msgs <-chan string) {} // receive-only
func send(msgs chan<- string)    {} // send-only
```

Channel

```
func main() {
  messages := make(chan string)
  go func() {
    messages <- "ping"
  }()
  msg := <-messages
  fmt.Println("got:", msg)
}
```

Channel

```
// any problem with this code?
func printNumber(numbers chan int) {
  for number := range numbers {
    fmt.Println("got:", number)
  }
}

func main() {
  numbers := make(chan int)
  go printNumber(numbers)
  go func() {
    for i := 1; i <= 5; i++ {
      numbers <- i
    }
  }()
}
```

sync.WaitGroups

WaitGroups allow us to wait until all goroutines within that **WaitGroup** have successfully executed

sync.WaitGroups

```
func main() {
  numbers := make(chan int)
  wg := sync.WaitGroup{}
  wg.Add(1)
  go func() {
    defer wg.Done()
    printNumber(numbers)
  }()
  // continue main.go
  wg.Add(1)
  go func() {
    defer wg.Done()
    for i := 1; i <= 5; i++ {
      numbers <- i
    }
    close(numbers)
  }()
  wg.Wait()
}
```

Live code

Assignment

Count the number of occurrences of a **word** in a folder containing multiple text files.

Requirements:

- Using goroutines & channels to solve it

References

Golangbot - Mutex

Effective Go - Concurrency

Concurrency is not parallelism

Share Memory By Communicating

Go Concurrency Patterns: Context

Go Concurrency Patterns: Pipelines and cancellation

Advanced Go Concurrency Patterns

Thanks