# **SESSION 1 ### Concurrence problem**

The code makes four concurrent web requests. But, as we've seen before, the race condition causes the program to exit before we have a response from them.

1. You have to wait until all request are done

2. Delete a character from one of the URLs. implement a way to cancel all the requests that are not completed yet when the error appears

# **1. You have to wait until all request are done.**

Solving using WaitGroups.

**Best practices**: using defer to notify on errors or not and simplify code.

package main

import (

    "io"

    "net/http"

    "os"

    "sync"

)

func main() {

    var wg sync.WaitGroup

    sites := []string{

        "https://www.google.com",

        "https://drive.google.com",

        "https://maps.google.com",

        "https://hangouts.google.com",

        "https://www.youtube.com",

        "https://meetup.com",

        "https://www.udc.es",

        "https://github.com/davorpa",

        "https://www.docker.com",

        "http://amazon.es",

        "https://twitter.com",

        "https://gobyexample.com",

    }

    for \_, site := range sites {

        wg.Add(1)

        go func(site string) {

            defer wg.Done()

            res, err := http.Get(site)

            if err != nil {

                io.WriteString(os.Stderr, site+"\t\t FETCH FAIL: "+err.Error()+"\n")

                return

            }

            io.WriteString(os.Stdout, site+"\t\t "+res.Status+"\n")

        }(site)

    }

    wg.Wait()

}

## **Run exit 1**

[Running] go run "e:\proyectos\cursos\curso-iniciacion-go-gdgmarbella\exercise\problem\requests.go"

11: https://gobyexample.com      200 OK

4: https://www.youtube.com       200 OK

0: https://www.google.com        200 OK

6: https://www.udc.es        200 200

8: https://www.docker.com        200 OK

10: https://twitter.com      200 OK

3: https://hangouts.google.com       200 OK

2: https://maps.google.com       200 OK

1: https://drive.google.com      200 OK

9: http://amazon.es      200 OK

5: https://meetup.com        200 OK

7: https://github.com/davorpa        200 OK

[Done] exited with code=0 in 2.516 seconds

## **Run exit 2**

[Running] go run "e:\proyectos\cursos\curso-iniciacion-go-gdgmarbella\exercise\problem\requests.go"

7: https://github.com/davorpa        200 OK

11: https://gobyexample.com      200 OK

0: https://www.google.com        200 OK

8: https://www.docker.com        200 OK

4: https://www.youtube.com       200 OK

6: https://www.udc.es        200 200

3: https://hangouts.google.com       200 OK

10: https://twitter.com      200 OK

2: https://maps.google.com       200 OK

1: https://drive.google.com      200 OK

9: http://amazon.es      200 OK

5: https://meetup.com        200 OK

[Done] exited with code=0 in 2.124 seconds

## **Run exit with unknown url**

[Running] go run "e:\proyectos\cursos\curso-iniciacion-go-gdgmarbella\exercise\problem\requests.go"

12: https://unkdnasd.google.com      FETCH FAIL: Get "https://unkdnasd.google.com": dial tcp: lookup unkdnasd.google.com: no such host

7: https://github.com/davorpa        200 OK

11: https://gobyexample.com      200 OK

0: https://www.google.com        200 OK

4: https://www.youtube.com       200 OK

8: https://www.docker.com        200 OK

6: https://www.udc.es        200 200

10: https://twitter.com      200 OK

3: https://hangouts.google.com       200 OK

2: https://maps.google.com       200 OK

1: https://drive.google.com      200 OK

9: http://amazon.es      200 OK

5: https://meetup.com        200 OK

[Done] exited with code=0 in 2.375 seconds

As we can see comparing runnings, the order of request execution is not warranted using this approach.

# **2. Delete a character from one of the URLs. implement a way to cancel all the requests that are not completed yet when the error appears.**

Solving using channels and cancellable contexts.

**Best practices**: using defer to notify on errors or not and simplify code and avoid thread leaks.

package main

import (

    "context"

    "io"

    "net/http"

    "os"

    "strconv"

    "sync"

)

func main() {

    var wg sync.WaitGroup

    // A cancelable context to allow comunicate goroutines one each other

    ctx, cancel := context.WithCancel(context.Background())

    // Best Practice: defer context `cancel` to avoid thread leaks

    defer cancel()

    respChan := make(chan bool)

    sites := []string{

        "https://www.google.com",

        "https://drive.google.com",

        "https://maps.google.com",

        "https://hangouts.google.com",

        "https://www.youtube.com",

        "https://meetup.com",

        "https://www.udc.es",

        "https://github.com/davorpa",

        "https://www.docker.com",

        "http://amazon.es",

        "https://twitter.com",

        "https://gobyexample.com",

        "https://unkdnasd.google.com", // don't exists so...

        "https://www.yahoo.com",      // this next url is not fetched

    }

    for idx, site := range sites {

        wg.Add(1)

        go func(site string, idx int, ctx context.Context, respChan chan<- bool) {

            // `defer` is a way to DRY the sync notification, on an http response error or not.

            // it force the execution after goroutine function has exit

            defer wg.Done()

            res, err := http.Get(site)

            if err != nil {

                // notify error response to channel

                respChan <- false

                io.WriteString(os.Stderr, strconv.Itoa(idx)+": "+site+"\t\t FETCH FAIL: "+err.Error()+"\n")

                return

            }

            io.WriteString(os.Stdout, strconv.Itoa(idx)+": "+site+"\t\t "+res.Status+"\n")

            // notify success response to channel

            respChan <- true

        }(site, idx, ctx, respChan)

        // test if channel notification fails

        if !<-respChan {

            // avoid fetch next urls

            cancel()

            break

        }

    }

    wg.Wait()

}

## **Run exit**

[Running] go run "e:\proyectos\cursos\curso-iniciacion-go-gdgmarbella\exercise\problem\requests.go"

0: https://www.google.com        200 OK

1: https://drive.google.com      200 OK

2: https://maps.google.com       200 OK

3: https://hangouts.google.com       200 OK

4: https://www.youtube.com       200 OK

5: https://meetup.com        200 OK

6: https://www.udc.es        200 200

7: https://github.com/davorpa        200 OK

8: https://www.docker.com        200 OK

9: http://amazon.es      200 OK

10: https://twitter.com      200 OK

11: https://gobyexample.com      200 OK

12: https://unkdnasd.google.com      FETCH FAIL: Get "https://unkdnasd.google.com": dial tcp: lookup unkdnasd.google.com: no such host

[Done] exited with code=0 in 8.453 seconds

The 13th request (<https://www.yahoo.com>) is not fetched due to 12th throws an error

Also as we can see, with this approach (using channels), the order of request execution is warranted in the same way as declared.