Netcentric lab 2

Nguyen Manh Viet Khoi
ITCSIU21081

Ex1

```go
package main

import (
   "fmt"
   "strings"
   "sync"
)

func count(word string, result chan<- map[rune]int, wg
*sync.WaitGroup) {
   defer wg.Done()
   counts := make(map[rune]int)
   for _, r := range word {
       counts[r]++
   }
   result <- counts
}

func main() {
   var wg sync.WaitGroup
   var str string = "we wjwelk weldk wlkedj wledj wel"
   // Add the number of goroutines to wait for

   wordChan := make(chan map[rune]int)
   // split the string by the space
   words := strings.Split(str, " ")
   wg.Add(len(words))
   fmt.Print(len(words), "\n")
```

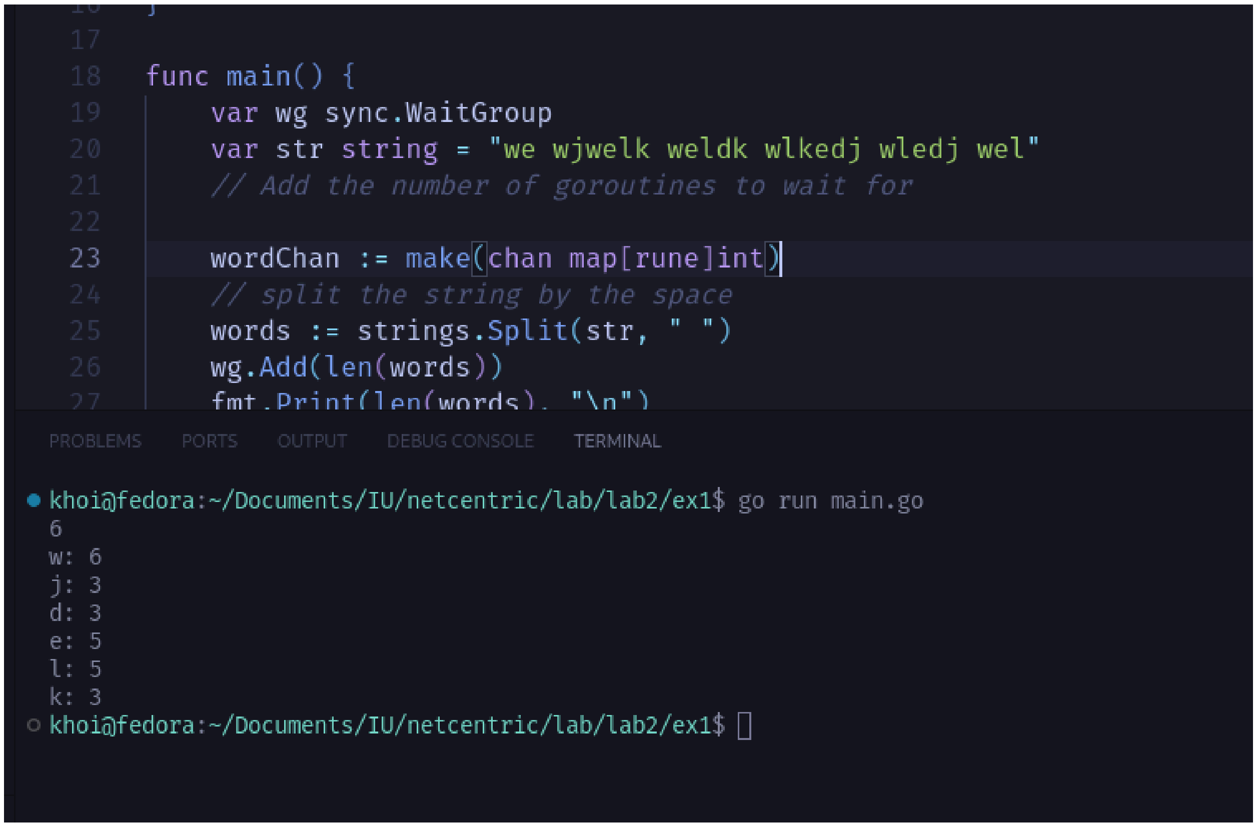
```go
    for _, word := range words {
        go count(word, wordChan, &wg)
    }

    // Wait for all goroutines to complete
    go func() {
        wg.Wait()
        close(wordChan)
    }()
    <-wordChan

    // combine the results
    result := make(map[rune]int)
    for word := range wordChan {
        for key, value := range word {
            result[key] += value
        }
    }

    // print the result
    for key, value := range result {
        fmt.Printf("%c: %d\n", key, value)
    }
}
```

Result



```go
func main() {
    var wg sync.WaitGroup
    var str string = "we wjwelk weldk wlkedj wledj wel"
    // Add the number of goroutines to wait for

    wordChan := make(chan map[rune]int)
    // split the string by the space
    words := strings.Split(str, " ")
    wg.Add(len(words))
    fmt.Print(len(words), "\n")
```

```
khoi@fedora:~/Documents/IU/netcentric/lab/lab2/ex1$ go run main.go
6
w: 6
j: 3
d: 3
e: 5
l: 5
k: 3
khoi@fedora:~/Documents/IU/netcentric/lab/lab2/ex1$
```

This Go program counts the frequency of each character in a string by splitting the string into words and processing each word concurrently using goroutines and channels. Here's a brief explanation:

- It defines a count function that counts the frequency of characters in a given word and sends the result to a channel.
- In the main function, it initializes a channel wordChan to collect the frequency counts of characters.
- The string str is split into words based on spaces.
- For each word, a goroutine is launched to count the frequency of characters using the count function.
- The main goroutine waits for all the counting goroutines to finish their work using a WaitGroup.
- After all counting goroutines have finished, the program combines the frequency counts from each word.

Finally, it prints the combined frequency counts of characters.
Overall, this program demonstrates concurrency in Go, efficiently counting the frequency of characters in a string using goroutines and channels.

Ex2

```go
package main

import (
   "fmt"
   "math/rand"
   "sync"
   "time"
)

const (
   maxStudents    = 30
   totalStudents  = 100
   maxReadingTime = 4
)

// Student represents a student in the library
type Student struct {
   id   int
   time int
}

func main() {
   // Create a channel to store students
   students := make(chan Student, maxStudents)
   // Create a wait group to wait for all students to finish
   var wg sync.WaitGroup

   // Create a goroutine for each student
   for i := 0; i < totalStudents; i++ {
       time.Sleep(time.Second)
```

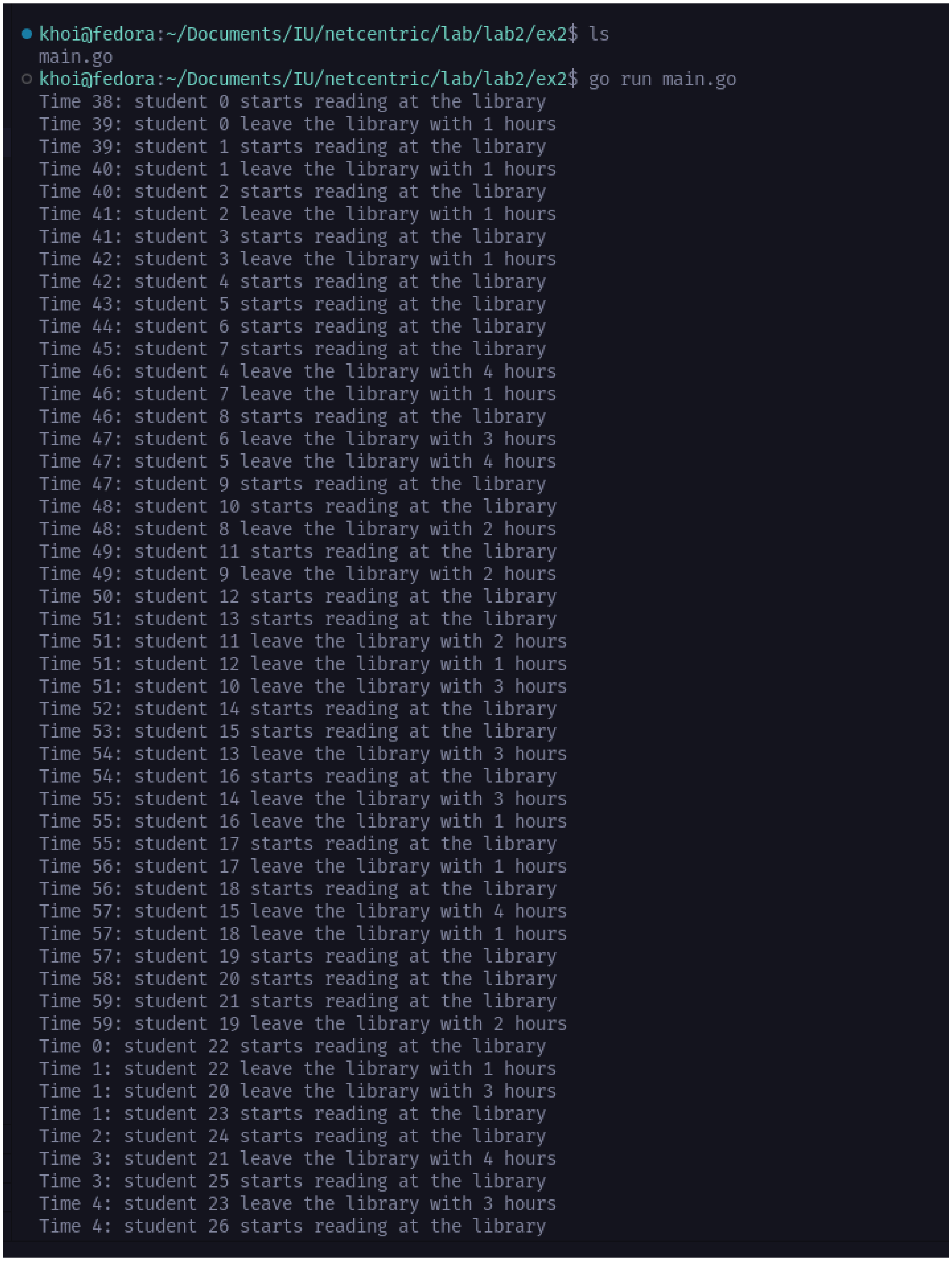
```
        duration := rand.Intn(maxReadingTime) + 1
        student := Student{
            id:   i,
            time: duration,
        }
        // Add the student to the wait group
        wg.Add(1)
        go func(s Student) {
            defer wg.Done()
            students <- s
            fmt.Printf("Time %d: student %d starts reading at the
library\n", time.Now().Second(), s.id)
            time.Sleep(time.Second * time.Duration(s.time))
            fmt.Printf("Time %d: student %d leave the library
with %d hours\n", time.Now().Second(), s.id, s.time)
            <-students
        }(student)
    }

    go func() {
        wg.Wait()
        close(students)
    }()

    fmt.Print("No more students in the library\n")

}
```

Result



```
● khoi@fedora:~/Documents/IU/netcentric/lab/lab2/ex2$ ls
  main.go
○ khoi@fedora:~/Documents/IU/netcentric/lab/lab2/ex2$ go run main.go
  Time 38: student 0 starts reading at the library
  Time 39: student 0 leave the library with 1 hours
  Time 39: student 1 starts reading at the library
  Time 40: student 1 leave the library with 1 hours
  Time 40: student 2 starts reading at the library
  Time 41: student 2 leave the library with 1 hours
  Time 41: student 3 starts reading at the library
  Time 42: student 3 leave the library with 1 hours
  Time 42: student 4 starts reading at the library
  Time 43: student 5 starts reading at the library
  Time 44: student 6 starts reading at the library
  Time 45: student 7 starts reading at the library
  Time 46: student 4 leave the library with 4 hours
  Time 46: student 7 leave the library with 1 hours
  Time 46: student 8 starts reading at the library
  Time 47: student 6 leave the library with 3 hours
  Time 47: student 5 leave the library with 4 hours
  Time 47: student 9 starts reading at the library
  Time 48: student 10 starts reading at the library
  Time 48: student 8 leave the library with 2 hours
  Time 49: student 11 starts reading at the library
  Time 49: student 9 leave the library with 2 hours
  Time 50: student 12 starts reading at the library
  Time 51: student 13 starts reading at the library
  Time 51: student 11 leave the library with 2 hours
  Time 51: student 12 leave the library with 1 hours
  Time 51: student 10 leave the library with 3 hours
  Time 52: student 14 starts reading at the library
  Time 53: student 15 starts reading at the library
  Time 54: student 13 leave the library with 3 hours
  Time 54: student 16 starts reading at the library
  Time 55: student 14 leave the library with 3 hours
  Time 55: student 16 leave the library with 1 hours
  Time 55: student 17 starts reading at the library
  Time 56: student 17 leave the library with 1 hours
  Time 56: student 18 starts reading at the library
  Time 57: student 15 leave the library with 4 hours
  Time 57: student 18 leave the library with 1 hours
  Time 57: student 19 starts reading at the library
  Time 58: student 20 starts reading at the library
  Time 59: student 21 starts reading at the library
  Time 59: student 19 leave the library with 2 hours
  Time 0: student 22 starts reading at the library
  Time 1: student 22 leave the library with 1 hours
  Time 1: student 20 leave the library with 3 hours
  Time 1: student 23 starts reading at the library
  Time 2: student 24 starts reading at the library
  Time 3: student 21 leave the library with 4 hours
  Time 3: student 25 starts reading at the library
  Time 4: student 23 leave the library with 3 hours
  Time 4: student 26 starts reading at the library
```

- This Go program simulates students visiting a library, where they spend a random amount of time reading before leaving. Here's a brief explanation:

    - It initializes a channel for students to store student objects representing those in the library.
    - It creates a WaitGroup to ensure that the program waits for all students to finish reading before exiting.
    - For each student (up to the total number specified), it generates a random reading duration and launches a goroutine to represent the student.
    - Each goroutine adds the student to the students channel, prints a message indicating the start of reading, sleeps for the specified duration, then prints a message indicating the student leaving.
    - If the channel is full of students then other coroutines have to wait until a new slot is available.
    - After all students are processed, the program closes the students channel and prints a message indicating that there are no more students in the library.

Overall, this program demonstrates basic concurrency in Go, using goroutines and channels to simulate multiple students accessing a shared resource (the library) concurrently.