Distributed Communication Homework

Li Jianhao
lijianhao288@hotmail.com

1 Homework

1.1 p1

Declare an uint64 **receivedCount**. Declare a channel of int **c**.

The main function creates two goroutines (We call it g1 and g2).

g1 send 0 till 11 to the channel **c**. For each sending, g1 will print out "g1 sent $< n >$". After sending the numbers, g1 do an operation to indicate that no more values will be sent to **c**.

g2 use a for range to receive messages from the channel **c**. For each received message, g2 print "g2 received $< n >$" and add 1 to the **receivedCount atomically** with the function in package "sync/atomic".

The main function **wait** for g1 and g2 and print out "Received count:" and **receivedCount**.

1.2 p2

Create a slice of int. Its name is **ints**. It has five initial elements: 1,2,-3,-4,5.

Create a function **handleInt** which takes an integer n and returns (n+6)*3 .

Create a map **intMap** which maps the int to string.

For each element in the **ints**, we start a new goroutine to write the result of **handleInt** as a string into the **intMap**. The key is the original int, the value is the result of **handleInt** as a string.

Use the Mutex to deal with the mutual exclusion problem when accessing the **intMap**.

Use the WaitGroup, let the main function wait until all the created goroutines finish. The main function will print out the **intMap** after that.

The output will be like:

```
map[-4:6 -3:9 1:21 2:24 5:33]
```