# CSC 456 Operating Systems Spring 2021

**Krebsbach**

**DUE: February 11th ,11:00 AM CST (dropbox) Assignment #2 - *Chapter 4* 15 points**

Please put completed assignment work into the dropbox.

# Do the following: (1 problems)

**1.** Exercise 4.11 possess the question “*Is it possible to have concurrency but not parallelism*” While investigating the programming language **GO** I came across a nice demonstration that not only addresses this question but also speaks more generally about the concept of Concurrency and that of Parallelism. Rob Pike has a nice set of slides that visually depicts the concepts of concurrency and parallelism in very nice way.

I found having the presentation open in one window and the slides in another helpful as you can move the slides along with his talk.

The slides we are interested in end around #29, after that it is some examples in the GO language which you do NOT need to watch. (the rest is interesting and you see several of the concepts from Ch 3 & 4 so when you have time  )

Pike’s Talk: <https://blog.golang.org/concurrency-is-not-parallelism>

Talk slides: <https://talks.golang.org/2012/waza.slide#1>

1. Please answer the question “*Is it possible to have concurrency but not parallelism*).

Yes you can have concurrency without parallelism. Concurrency is a composition of independently executing. Parallelism is simultaneously executing multiple things, related or not. Concurrency does not equal simultaneity in this situation.

1. How would Pike answer the question “**Why is it important to think of problems and their solutions in a concurrent way and not in a parallel way**”? So, why is that approach better even if the solution eventually ends up being executed in parallel?

* “Concurrency is the expression of the problem, and you parallelize it by instantiating more instances of that solution.” (concurrent composition of processes)
* “Adding things to the design can make it more efficient, and parallelism can come from better concurrent expression of the problem.”
* “… I might only be able to run one [thread] at a time, in which case this design would only run at the rate of a single [processor] like the original problem and the other seven would all be idle while [it’s] running; but the design is still correct. And that’s a pretty big deal because it means that we don’t have to worry about parallelism when we are doing concurrency because if we get the concurrency right, then the parallelism is actually a free variable that we can decide just how many [threads] are busy.”