Chapter 6

Concurrency

process

- 어떠한 한 프로그램에 대한 인스턴스다.
- 운영체제로 부터 프로그램이 동작할 수 있는 재원을 할당 받는다.
- 모든 프로세스는 **하나 이상의 스레드** 를 갖는다.

thread

- **프로세스** 내에서 동작
- 프로세스를 위해 처음 생성된 스레드를 **`main thread`** 라 부른다.
- **`main thread`** 가 종료되면 프로그램이 종료된다.

병렬성

- 각각의 프로세스에서 동작을 수행.
- 두 개의 책상에서 두사람이 각기 다른 일을 수행.
- 동일한 시간에 여러 작업을 수행.

동시성

- 하나의 프로세스에서 여러 동작을 수행.
- 한 책상에서 한 사람이 여러일을 수행.
- 동일한 시간에 여러작업은 불가. (특정 시간에 한 작업만 가능)
- **Go** 에서 동시성이란 함수들이 다른 함수들과 서로 독립적으로 실행할 수 있는 기능을 의미.

Label (for)

- **Label**을 사용하면 **for**문에 이름을 부여하여 특정 **for**문으로 이동할 수 있다.

```
func ExecuteLabel1() {
FOR_LOOP1: // Label 다음에는 반드시 이름을 붙이고자 하는 for문이 와야 한다.
    for outer := 0; outer < 10; outer++ {
        for inner := 0; inner < 10; inner++ {
            if outer == inner {
                continue FOR_LOOP1 // FOR_LOOP1로 되돌아 간다.
            } else {
                fmt.Printf("(%d : %d)\n", outer, inner)
            }
        }
    }
}
```

Label (goto)

- goto문을 사용할 때에도 Label을 사용할 수 있다.

```
func ExecuteGoto(casenum int) {
    if casenum == 0 {
        goto CASE1
    }
    if casenum == 1 {
        goto CASE2
    }
    CASE1:
    fmt.Println("ZERO CASE!")

    CASE2:
    fmt.Printf("ONE CASE!(%d)", casenum)
}
```

goroutine

- 경량화 된 스레드. (**2kb**, 일반적인 스레드는 보통 최고 **2mb**)
- **func main()** 함수는 **main goroutine**에서 실행된다.
- 실행하고자 하는 함수 앞에 **go** 를 추가하면 **goroutine** 으로 실행된다.

```
func executeFunc() {
    // do something
}

go executeFunc() // goroutine 으로 함수 실행
```

GOMAXPROCS (runtime package)

- 고루틴을 실행하는 논리 프로세서의 수를 알거나 설정할 수 있다.
- `runtime.GOMAXPROCS(n int)`는 이전 설정 상태값을 반환한다.
- `n < 1` 인 경우, 현재 설정을 변경하지 않는다. 즉, 0을 입력하면 현제 설정상태를 알 수 있다.
- **별도로 값을 설정하지 않으면 CPU 코어 갯수로 설정된다. (v1.7 현재)**

```
import "runtime"

runtime.GOMAXPROCS(2) // 프로세스를 2로 설정하고, 이전 상태를 리턴한다.
// 0을 입력하면 설정을 변경하지 않고 이전상태를 반환한다.
// 즉, 현재 설정된 값을 알 수 있다.
runtime.GOMAXPROCS(0)
```

WaitGroup (sync package)

- 고루틴이 실행되면 고루틴이 끝나기 전에 **main**함수가 끝나는 경우가 발생.
- 따라서 고루틴이 끝날때 까지 기다릴 수 있는 기능이 필요.
- **sync.WaitGroup**을 사용하여 특정 갯수만큼 지정하여 그 갯수만큼 기다릴 수 있음.

```
import "sync"

var wg sync.WaitGroup

wg.Add(1) // 고루틴의 갯수를 추가한다.
wg.Wait() // 입력된 고루틴이 끝날때까지 기다린다.
wg.Done() // 고루틴은 끝날때마다 해당 함수를 호출하여 WaitGroup에 알린다.
```