UCI Division of Continuing Education

Module 4: Threads in Go
Topic 3.1: Once Synchronization

Synchronous Initialization

Initialization

- must happen once
- must happen before everything else

- How do you perform initialization with multiple goroutines?
- Could perform initialization before starting the goroutines

Sync.Once

- Has one method, **`once.Do(f)`**
- Function `f` is executed only one time
  - Even if it is called in multiple goroutines
- All calls to `once.Do()` block until the first returns
  - Ensures that initialization is executes first

Sync.Once Example

- Make two goroutines, initialization only once
- Each goroutine executes `dostuff()`

```
var wg sync.WaitGroup

func main() {
   wg.Add(2)
   go dostuff()
   go dostuff()
   wg.Wait()
}
```

UCI Division of Continuing Education

Using Sync.Once

- `setup()` should execute only once
- "hello" should not print until `setup()` returns

```
var on sync.Once
func setup() {
   fmt.Println("Init")
}
func dostuff() {
   on.Do(setup)
   fmt.Println("hello")
   wg.Done()
}
```

UCI Division of Continuing Education

Execution Result

`Init`	Result of `setup()`
`Hello`	Result of one goroutine
`hello`	Result of the other goroutine

- **`Init`** appears only once
- **`Init`** appears before **`hello`** is printed