Ejercicios concurrencia en Go: gorutinas y canales

Sistemas Distribuidos.
Grado en Ing. en Telemática, 2019-2020.
Escuela Técnica Superior de Ingeniería de Telecomunicación.
Universidad Rey Juan Carlos.

---

1. Objetivos

- Comprender mejor el funcionamiento de las *'goroutines'* explicadas en clase.
- Comprender los problemas de concurrencia asociados al uso de *goroutines*.
- Comprender la utilización de *channels* para comunicación y sincronización entre *goroutines*.

Los ejemplos están tomados del capítulo 8 del libro "The Go Programming Language", indexado en la bibliografía esencial de la asignatura.

2. Ejemplo 1: Gorutina básica

Ejecuta el siguiente comando:

```
usuario@machine:~$ go get gopl.io/ch8/spinner
```

Abre con un editor (por ejemplo Atom) el archivo `$GOPATH/src/gopl.io/ch8/spinner/main.go`. Ejecuta el programa y responde a las siguientes preguntas:

- ¿Cuántas gorutinas se lanzan en este código, incluyendo la que ejecuta el hilo principal del programa?
- ¿Cómo puede mostrarse el símbolo de spinner girando en la pantalla del terminal aunque se esté ejecutando el cálculo de 45º número de Fibonnaci?
- ¿Por qué se deja de ver al spinner girando cuando el programa muestra el resultado con el número de Fibonacci solicitado?

3. Ejemplo 2: Clock server concurrente

Ejecuta el siguiente comando:

```
usuario@machine:~$ go get gopl.io/ch8/clock1
```

Abre con un editor (por ejemplo Atom) el archivo `$GOPATH/src/gopl.io/ch8/clock1/main.go`. A continuación ejecuta el programa y realiza las siguientes actividades para contestar las preguntas:

1. Una vez lanzado el programa `'clock1'`, utiliza el comando `netcat` para conectarte con el servidor que escucha en el puerto TCP 8000. Puedes usar el comando:

```
usuario@machine:~$ nc localhost 8000
```

2. Ahora, abre otra ventana de terminal de Linux (u otra pestaña en la misma ventana de terminal) y lanza dos procesos clientes de nuestro servidor de hora simultáneamente. ¿Es capaz de atender el servidor a los dos clientes a la vez? ¿Por qué?

3. Modifica el programa anterior para que se garantice que atiende a dos o más clientes de forma simultánea.

4. Ejercicio: relojes con zonas horarias

1. El programa `go get gopl.io/ch8/netcat1` muestra cómmo programar un emulador sencillo del programa `netcat` en lenguaje Go, usando la función `net.Dial`.

2. Modifica el programa anterior del servidor de hora concurrente, de forma que acepte como parámetro de entrada por línea de comandos un número de puerto.

3. Ahora, escribe un programa cliente `clockwall` que actúe como cliente de varios servidores de hora, leyendo la hora de cada uno y mostrándola por pantalla como si se tratase de los relojes con varias zonas horarias simultáneas que se encuentran en algunas oficinas de empresas. Puedes ejecutar varios servidores de hora que simulen zonas horarias ficticias para darle credibilidad.

```
usuario@machine:~$ TZ=US/Eastern $GOPATH/bin/clock2 -port 8010 &
usuario@machine:~$ TZ=Asia/Tokyo $GOPATH/bin/clock2 -port 8020 &
usuario@machine:~$ TZ=Europe/London $GOPATH/bin/clock2 -port 8030 &
usuario@machine:~$ $GOPATH/bin/clockwall NewYork=localhost:8010\
 Tokyo=localhost:8020 London=localhost:8030
```