

# Concurrencia en Go

IC-4700: Lenguajes De Programación

II Semestre, 2023

William Alfaro Quirós – 2022437996  
Jesus Cordero Díaz – 2020081049  
Grupo 20 - Prof. María Auxiliadora Mora



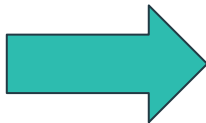
# Introducción



## Características:

- Simplicidad y eficiencia
- Concurrencia sencilla
- Gestión de paquetes sólida
- Sintaxis y formato sencillos

**Relevancia en la programación actual**



## Usos:

- Sistemas y redes
  - Big data
  - Aprendizaje automático
  - Edición de audio y video
  - Aplicaciones web
- 
- Lenguaje popular y en crecimiento
  - Adecuado para una amplia gama de aplicaciones
  - Fácil de aprender y usar

# Concurrencia

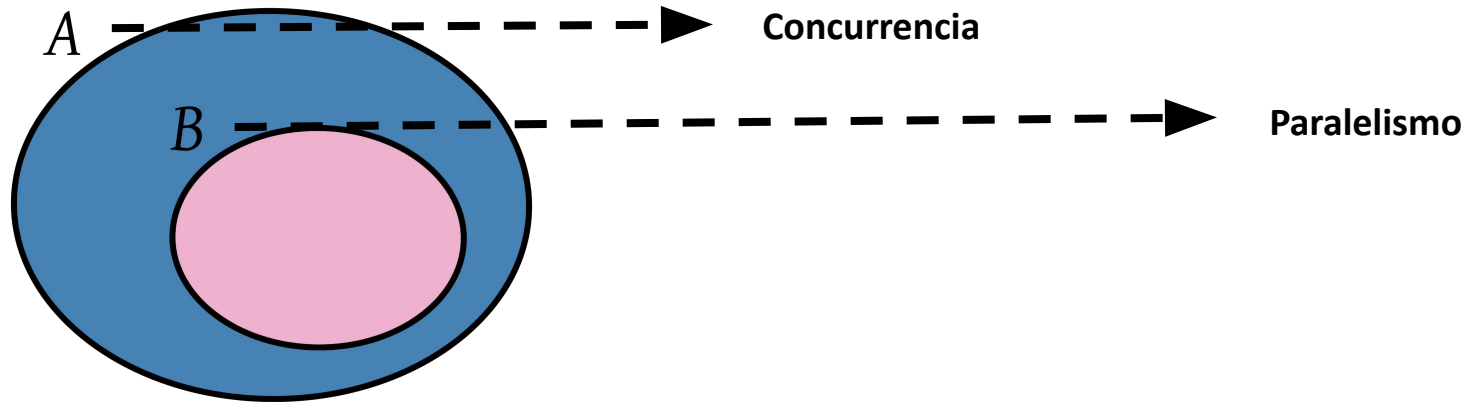
Capacidad de un programa para ejecutar múltiples tareas simultáneamente, o al menos, para organizarlas de manera que parezca que se ejecutan en paralelo.

**Go** → Lenguaje de programación que tiene un soporte nativo para la concurrencia.

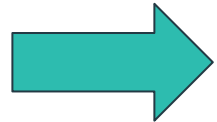
## Mecanismos de concurrencia en Go

- **Goroutines:** son unidades de ejecución ligeras que se pueden crear y ejecutar de forma independiente.
- **Canales:** son estructuras de datos que se utilizan para comunicar datos entre goroutines.
- **Select:** se utiliza para controlar el flujo de ejecución entre goroutines.

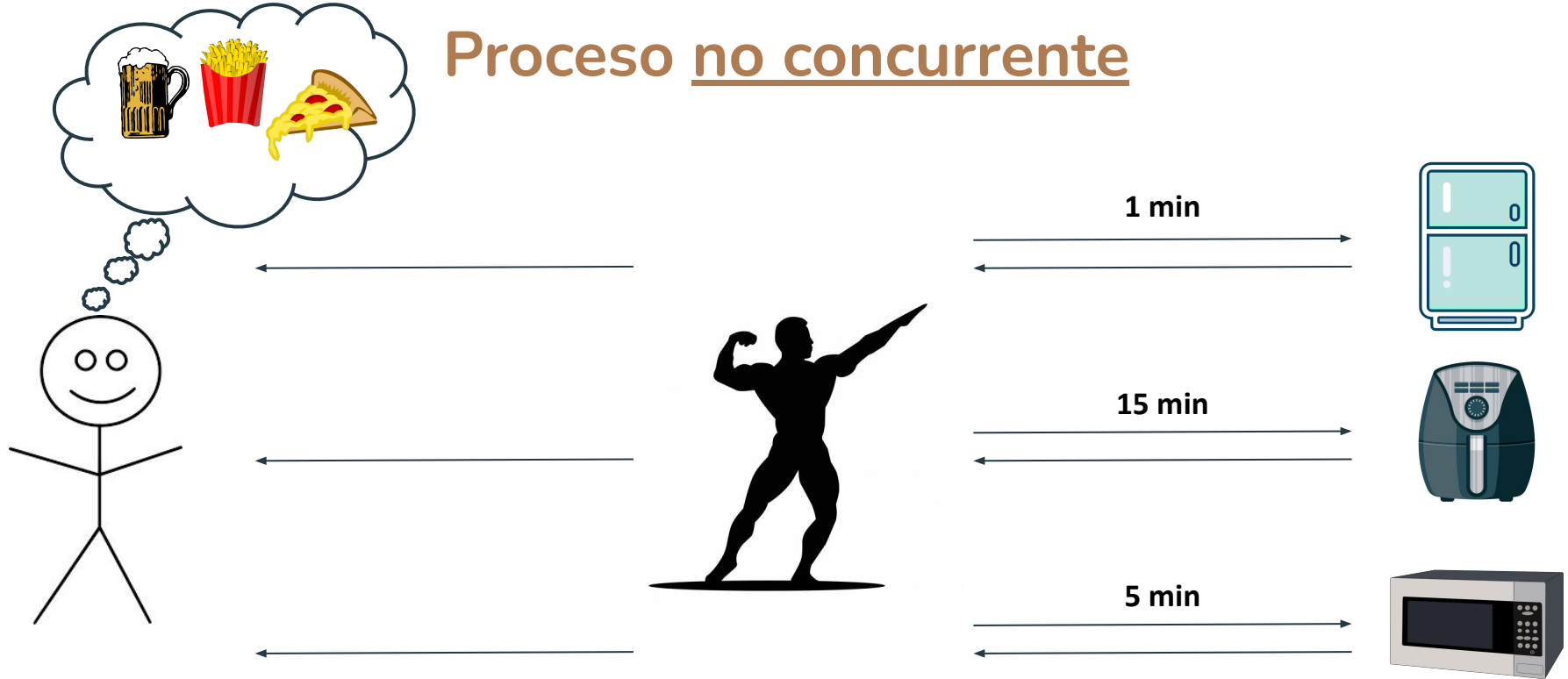
# Entendamos abstractamente la concurrencia



Ejemplos

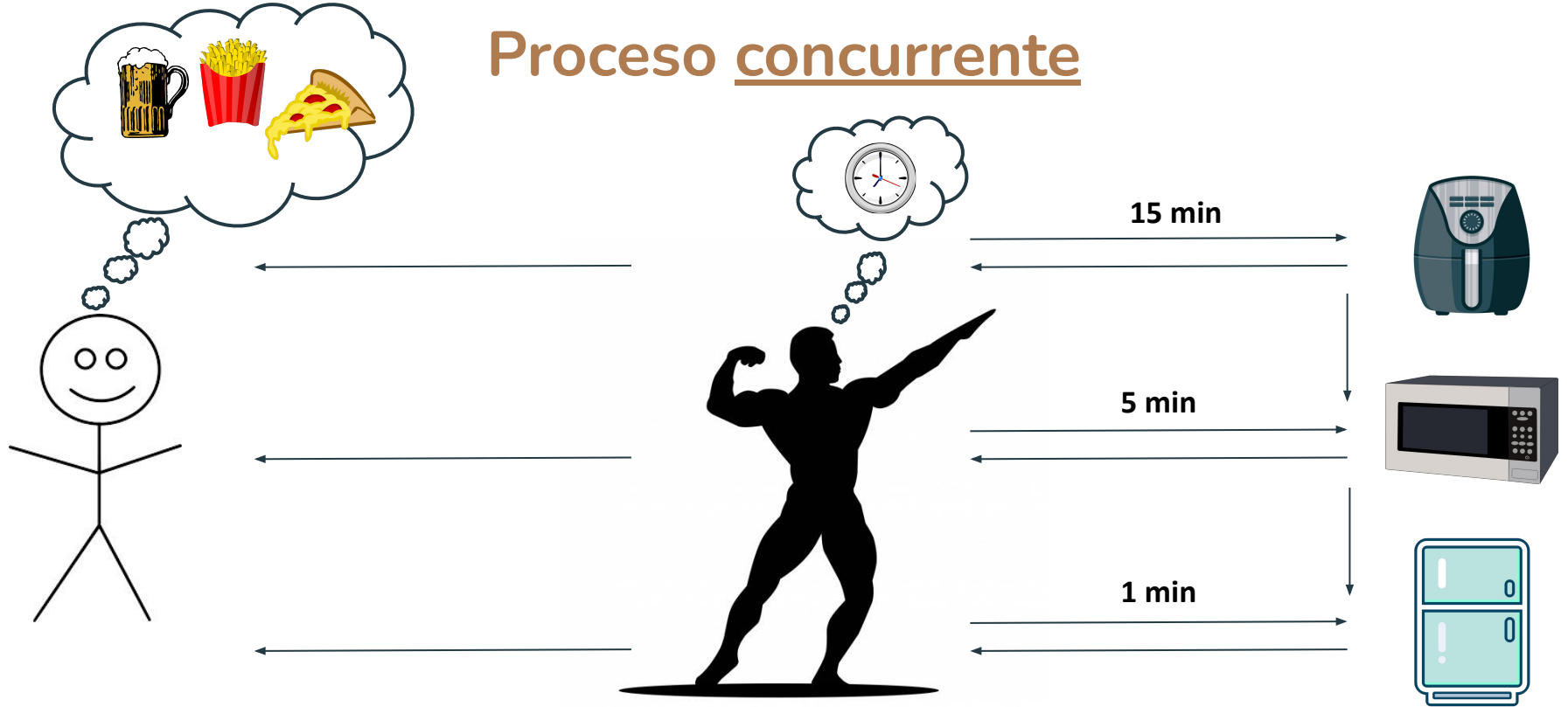


# Proceso no concurrente



Tiempo = 21 min

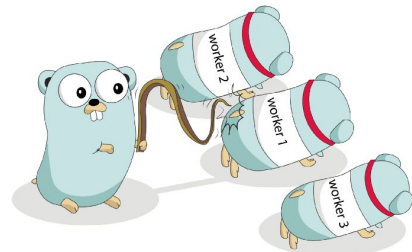
# Proceso concurrente



Tiempo = 15 min

# “Goroutines”

- Son como hilos ligeros que permiten la ejecución concurrente de tareas en un programa.
  - Son administradas por el propio sistema de tiempo de ejecución de Go,
- 
- Se crean fácilmente utilizando la palabra clave **'go'** seguida de una función.
  - Cuando se llama a una función con **'go'** delante, se ejecuta como una Goroutine de forma independiente y concurrente.
  - Facilitan la programación concurrente y paralela en Go sin la complejidad de la gestión manual de hilos.



# Ejemplo 1: Simulación de tareas concurrentes

```
C: > Users > alfar > OneDrive - Estudiantes ITCR > Desktop > Codigos_Ejemplo.go > main
1  package main
2
3  import (
4      |   "time"
5      | )
6
7  func tarea1() {
8      |   // Lógica de la tarea 1
9      | }
10
11 func tarea2() {
12     |   // Lógica de la tarea 2
13     | }
14
15
16 func main() {
17     |   go tarea1()
18     |   go tarea2()
19     |   time.Sleep(time.Second) // Esperar para que las Goroutines terminen
20 }
```



## Ejemplo 2: Descarga concurrente de páginas web

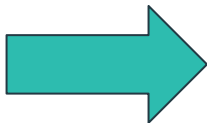
C: > Users > alfar > OneDrive - Estudiantes ITCR > Desktop > Codigos\_Ejemplo.go > descargarPagina

```
1 package main
2
3 import "time"
4
5 func main() {
6     urls := []string{"https://sitio1.com", "https://sitio2.com", "https://sitio3.com"}
7
8     for _, url := range urls {
9         go descargarPagina(url)
10    }
11
12    time.Sleep(time.Second * 5) // Esperar a que todas las descargas terminen
13 }
14
15 func descargarPagina(url string) {}
16     // Lógica para descargar la página web
17 }
```

# Canales

- Diseñados para facilitar la sincronización y la comunicación entre goroutines,
- Proporcionan una forma segura y eficaz de transmitir datos entre goroutines
- Permiten comunicación y coordinación de acciones de manera eficiente.

Ejemplo



```
Codigos_Ejemplo.go > main
1  package main
2
3  import (
4      "fmt"
5  )
6
7  func main() {
8      // Crear un canal de tipo entero
9      canal := make(chan int)
10
11     // Goroutine que envía datos al canal
12     go func() {
13         for i := 1; i <= 5; i++ {
14             canal <- i // Envía números del 1 al 5 al canal
15         }
16         close(canal) // Cerrar el canal cuando se complete la tarea
17     }()
18
19     // Goroutine que recibe datos del canal
20     go func() {
21         for numero := range canal {
22             fmt.Printf("Número recibido: %d\n", numero)
23         }
24     }()
25
26     // Esperar a que ambas goroutines terminen
27     <-canal
28     <-canal
29 }
```

# Concurrencia en Go


Ventajas

VS


Desventajas

- Escalabilidad
- Eficiencia
- Sencillez
- Comunicación entre goroutines
- Paralelismo

- Complejidad
- Race Conditions
- Goroutines olvidadas:
- Dificultad en la depuración:
- Overhead

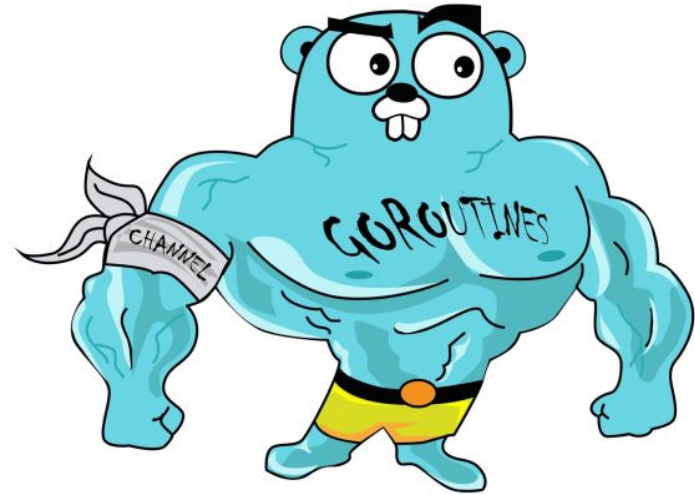


**Ejemplo:**  
Programa para escribir  
números del 0 al FFFFFFFF



# Consideraciones Finales

- Escalabilidad
- Eficiencia
- Resistencia y Tolerancia a Fallos
- Mejora del Rendimiento
- Experiencia del Usuario
- Competitividad



# Referencias

Chio Code. (2020). Concurrency en Go | Concepto e Introducción. YouTube.

[https://www.youtube.com/watch?v=T\\_NlyqVQrx4&ab\\_channel=ChioCode](https://www.youtube.com/watch?v=T_NlyqVQrx4&ab_channel=ChioCode)

Hdeleon.net. (2023). ¿Qué Diablos es PROGRAMACIÓN CONCURRENTE?. YouTube.

[https://www.youtube.com/watch?v=jw\\_vraxfnQw&ab\\_channel=hdeleon.net](https://www.youtube.com/watch?v=jw_vraxfnQw&ab_channel=hdeleon.net)

Walker, R. (2023). Concurrency en Go. AppMaster - ultimate all-in no-code platform.

<https://appmaster.io/es/blog/concurrency-ir>