|  |  |
| --- | --- |
| http://www.unla.edu.ar/sistemas/sls/imagenes/logo-UNLa.jpg | **LICENCIATURA EN SISTEMAS** Departamento de Desarrollo Productivo y Tecnológico Cátedra: Programación Concurrente |

|  |  |
| --- | --- |
|  |  |
|  |  |

**Documentación del Trabajo Practico Cuatrimestral**

**Integrantes:**

* Furszyfer Matías
* Guastadisegni Federico
* **Introducción y conceptos de la concurrencia:**

El concepto fundamental de la programación concurrente es la noción de Proceso.

**Proceso:** Cálculo secuencial con su propio flujo de control.

La concurrencia en software implica la existencia de diversos flujos de control en un mismo programa colaborando para resolver un problema.

* **Los procesos vs los hilos:**

En el contexto del Sistema Operativo, un Proceso es una instancia de un Programa que está siendo ejecutado en el ordenador.

Proceso = Código de programa + Datos + Recursos

Un S.O. admite concurrencia si es capaz de manejar diversos procesos simultáneamente.

En el contexto de un Programa concurrente, un Hilo (Thread) es cada uno de los flujos secuenciales de control independientes especificados en el programa.

* **Procesos Secuenciales**

Un proceso tradicional, correspondiente a un programa secuencial, es un proceso que posee un único hilo de control.

El contador de programa, el estado de la pila y de los objetos definen el estado del sistema.

* **Procesos concurrentes**

Un programa concurrente da lugar, durante su ejecución, a un proceso con varios hilos de ejecución.

Por ejemplo, un sistema concurrente es uno que utilice el protocolo cliente/servidor, teniendo muchos procesos ejecutándose al mismo tiempo:

* **Concurrencia Software vs. Paralelismo Hardware**

La concurrencia software es un concepto lógico, no implica la existencia de paralelismo en el hardware:

1. Las operaciones hardware ocurren en paralelo si ocurren al mismo tiempo.

2. Las operaciones (software) en un programa son concurrentes si pueden ejecutarse en paralelo, aunque no necesariamente deben ejecutarse así.

**Tipos de concurrencia**

**Concurrencia Física:** Existe más de un procesador y varias unidades (hilos) de un mismo programa se ejecutan realmente de forma simultánea.

**Concurrencia Lógica:** Asumir la existencia de varios procesadores, aunque no existan físicamente. El implementador de tareas del lenguaje se encargará de “mapear” la concurrencia lógica sobre el hardware realmente disponible.

La concurrencia lógica es más general, pues el diseño del programa no está condicionado por los recursos de computación disponibles.

* **Arquitecturas Multiprocesador: Revisión histórica**

1. Años 50: Primeras máquinas con varios procesadores: Un procesador de propósito general y varios procesadores para controlar la E/S, Permitían ejecutar un programa mientras se realizaba E o S para otros programas. No se puede hablar de ejecución concurrente de programas.

2. Años 60 (inicio): Primeras multiproc. Reales: El programador de tareas del SO distribuía programas entre los procesadores a partir de una cola de trabajos. Permitía la concurrencia a nivel programa.

3. Años 60 (mitad): Ordenadores con hardware específico múltiple: Multiplican el número de multiplicadores en coma flotante o unidades aritméticas en coma flotante completas. Estas unidades ejecutan instrucciones procedentes de un único flujo de instrucciones (programa). Los compiladores determinaban qué instrucciones podían ejecutarse concurrentemente.

* **Concurrencia en un Programa:**

Desde el punto de vista de la programación, interesa la concurrencia lógica que puede existir en el interior de un programa. Un Lenguaje de Programación será concurrente si posee las estructuras necesarias para definir y manejar diferentes tareas (hilos de ejecución) dentro de un programa. El compilador y el SO serán los responsables de “mapear” la concurrencia lógica del programa sobre el hardware disponible.

**Lenguaje utilizado en nuestro programa : Go languague**

**Introducción al lenguaje:**

Go es un lenguaje de programación concurrente inspirado en la sintaxis de C. Ha sido desarrollado por Google. El fuerte de este lenguaje es que está diseñado específicamente para aprovecharlos distintos núcleos de un procesador. Tiene de forma nativa shared memory, que es un área de memoria compartida por todos los procesos que se están ejecutando en la maquina.

Actualmente sólo está disponible para los sistemas operativos Windows, GNU/Linux y Mac OS X.2 3

**Go Routines :** Una función de Go que permite ejecutar otra función de manera paralela alprograma principal.

**Channel :** Los channel proveen comunicación y sincronización entre las Go Routines.

**Multiplexación:** los canales son valores de “Primer tipo”, es decir, que pueden ser enviados a través de otros canales. Esta Propiedad hace que sea fácil de escribir un servicio multiplexador dado que, el cliente puede proporcionar, juntos con la petición, el canal al que debe responder.

**Explicación de nuestro programa**

**Concepto general:** Nuestro programa utiliza la concurrencia para poder ejecutar distintos hilos al mismo tiempo, cada uno de los cuales posee un dato tipo String, un retraso (Time.Duration) y el canal para poder realizar el bloqueo/desbloqueo de las goroutines. En este caso, lo usamos para saber cuando esta "lista" cada bebida.

* Explicaciones del código y su funcionamiento:

Para los **import** de nuestro programa, utilizamos “fmt” para realizar los Println, “time” para realizar el control del Sleep y para poder imprimir al final un tiempo de ejecución que tiene que ser acorde al retardo que nosotros elegimos .

La función “**ready**” recibe como parámetro el channel y el nombre de la bebida y el tiempo de retraso, imprimiendo “está listo” , pasando el valor al channel para que este se bloquee hasta que sea leído y pueda imprimir “está terminando!”

En el **main**, se toma el tiempo al inicio y al final de la ejecución con las variables t0 y t1 utilizando otra de las características de time que es el Now.

Se crea un **channel** tipo String.

Utilizando la función GO para indicar que es una **Goroutine**, se le pasa a ready el canal y el nombre de la bebida, por lo que el canal se va a bloquear.

Se imprime por pantalla **“Estoy esperando...”** que va a ser lo primero que se imprima, ya que el main es un hilo mas, y es el primero que se ejecuta. Luego con los sleep en la función ready, se le transfiere el control momentáneamente al hilo correspondiente.

Se utilizan las variables a, b y c para leer los valores del canal, desbloqueando el mismo, por lo que se imprime el nombre de las bebidas, y, como ya se desbloqueo, se imprime **“Esta terminando!”.**

Posteriormente se imprime el tiempo de ejecución.

En el caso que se le dé el mismo tiempo a todos los hilos, no siempre se van a ejecutar en el mismo orden, ya que esto depende de cómo el sistema ordena su schedule en ese momento.

Para poder ejecutar el programa, se debe tener instalado GO en el sistema, y se tiene que utilizar los comandos Go build y Go run para poder compilar y ejecutar el mismo.

**Problemas que encontramos**

* Primero que nada, adecuarnos a la sintaxis de un nuevo lenguaje (a pesar de ser similar a C)
* Comprender el concepto de concurrencia, y el cómo aplicarlo al lenguaje Go , que lo resolvimos investigando por internet.
* Como instalar y hacer correr el lenguaje en el sistema operativo, en particular en Linux. También pudimos solucionarlo investigando en internet.
* Como utilizar el Time.Sleep para poder "dormir" los distintos hilos, y el Time.Duration para establecer el tiempo que esos hilos van a estar dormidos. Nuestro principal problema fue con el tipo de dato que acepta el Time.Duration, que es int64. Lo solucionamos investigando en la documentación que está en la página de Go.
* Como utilizar los Channels, y como saber cuándo se bloquea/desbloquea. Lo solucionamos también con la documentación en la página de Go.
* Como calcular e imprimir el tiempo de ejecución del programa (que en este caso en particular, va a tener que ser el tiempo que nosotros le pasemos a las goroutines..). Solucionado investigando por internet.

**Conclusiones**

A pesar de ser un programa simple, muestra la concurrencia entre los distintos hilos, ya que si se le pone a todos el mismo tiempo, estos se ejecutan siempre de distinta manera, según el schedule del sistema operativo.

**Capturas del programa:**







