

UNIVERSIDAD EMPRESARIAL

SIGLO 21



CARRERA: LIC. EN INFORMATICA

Trabajo Práctico número 1

Materia: Algoritmos concurrentes y paralelos

Profesor: PIRAY, EDUARDO ENRIQUE

Modulo: 1

Alumno: Soler, Diego Francisco.

Fecha: 10/06/2024

Objetivos:

- Crear y manipular programas concurrentes en C++.
- Implementar programas concurrentes que requieran cierta sincronización.
- Adquirir conocimiento empírico del funcionamiento de threads en C++.

Consigna 1

Basado en la situación problemática, se debe analizar el comportamiento de los algoritmos presentados, estudiando sus particularidades desde el punto de vista de la implementación. Para llevar a cabo esta tarea es necesario ejecutar los algoritmos y luego de realizar diversas ejecuciones, documentar y explicar el comportamiento de tales programas.

Consigna 2

Con base en el enfoque presentado en el algoritmo 2, desarrollar una nueva versión del programa en el que se ejecuten 15 procesos, de manera que el proceso i -ésimo se pause un tiempo aleatorio (definir un rango en milisegundos que pueda ser perceptible), y que por pantalla se muestre el mensaje correspondiente ("Soy el proceso 1", "Soy el proceso 2", ... , "Soy el proceso n ") un número aleatorio de veces.

Consigna 3

Tal como se puede observar, los algoritmos anteriores son claros cuando el comportamiento del proceso puede estar encapsulado en una única función. Pero termina siendo confuso cuando los procesos necesitan llevar a cabo tareas de mayor complejidad. Para ello, un enfoque más general consiste en encapsular (en una clase de objetos) el comportamiento de un proceso, creando las instancias necesarias, y de este modo lanzar la función deseada del objeto en un *thread*. Analice de qué manera podría implementar este enfoque, describa las características y esboce una implementación del algoritmo.

Algoritmo 1:

El algoritmo uno lo que hace es crear 3 hilos, cada uno con un mensaje, delay y cantidad de repeticiones distintos, cada uno de estos hilos ejecutan la función misaludo de manera independiente respetando sus respectivos delays. Estos hilos están unidos mediante la función join, creando un hilo principal permitiendo asegurar que los 3 hilos terminen de ejecutarse antes de hacer un print del string "Fin".

Ahora procedemos a ejecutar el algoritmo 6 veces para testear su funcionamiento:

Test 3

[illegible][illegible][illegible]

Test 6

[illegible]

Página 4 | 7

Consigna 2

Para esta consigna vamos a modificar el código del algoritmo 2 para cumplir con la consigna, sumado a que le vamos a permitir al usuario que elija que proceso desea pausar, este proceso será pausado por única vez por 600 milisegundos y luego volverá a su funcionamiento normal. Además, creamos la función `generarAleatorio` para generar números aleatorios en base a un rango dado, modificamos la función `saludo` para que cumpla con la pausa mencionada y agregamos distintos loops para poder automatizar el proceso.

Todo el código se puede observar desde el archivo `consigan2.cpp`

Consigna 3

Ahora para la consigna 3 vamos a modificar el código de la consigna 2 pero utilizando programación orientada a objetos. En nuestro código vamos a crear la clase `proceso`, con los atributos `id` y `procesoPausar` (determina si es el proceso que se debe apusar). Esta clase tiene los métodos:

- `generarAleatorio`: Que genera un numero aleatorio dentro de n rango dado
- `saludo`: crea una variable que llama al método `generarAleatorio`, contiene un loop para imprimir los mensajes y el if que determina si se trata del hilo a ser pausado y si ya fue pausado una vez.

Por último, tenemos la función `main`, que crea una instancia de la clase `proceso` para cada hilo, hace el `join` para esperar a que todos los hilos terminen y por último imprime el string "Fin".

Todo el código se puede observar desde el archivo `consigan3.cpp`

Dificultades que me ocurrieron:

- En la consigna 2 y 3, cometí el error de querer usar el mismo loop para lanzar los hilos y hacer el join, generando que no ocurra la concurrencia dado que no se lanzaba otro hilo hasta que no terminara el anterior. Esto lo solucione separando los loops.
- Agregue un validador para el cin para que el usuario solo pueda ingresar un int del 1 al 15.