|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado por:** | **Revisado por:** | **Autorizado por:** | **Vigente desde:** |
|  |  | **Dr. Francisco Javier Solorio Ordaz** | **2 de agosto de 2019** |

**Manual de prácticas de**

**Temas Selectos de Programación I y II**

Índice de prácticas

Práctica 1: Tema 1 x

Práctica 2: Tema 2 xx

Práctica 3: Tema 3 xx

Práctica 4: Tema 4 xx

Práctica 5: Tema 5 xx

**Práctica #X**

**Programación concurrente**

***1. Seguridad en la ejecución***

|  |  |  |
| --- | --- | --- |
|  | **Peligro o Fuente de energía** | **Riesgo asociado** |
| 1 | Tensión alterna | Electrocución |

***2. Objetivos de aprendizaje***

OBJETIVO GENERAL: Entender el concepto de programación concurrente y aplicarlo a su conveniencia en proyectos de diversa naturaleza.

OBJETIVOS ESPECÍFICOS:

* Realizar un programa que implemente hilos en su ejecución.
* Evaluar los tiempos de ejecución de programas secuenciales en comparación con programas distribuidos.
* Usar mecanismos para sincronizar el acceso de hilos a objetos en un programa.
* Entender el alcance de esta tecnología en el desarrollo de proyectos de cualquier índole.

***3. Introducción***

Un programa concurrente es uno en el que un cálculo puede avanzar sin esperar a que se completen todos los demás cálculos. Su desarrollo fue motivado originalmente por el deseo de desarrollar sistemas operativos más confiables, aunque actualmente su alcance ha llegado a muchas áreas de desarrollo tecnológico.

Para entender esto más a detalle, es necesario introducir el concepto de algoritmo y proceso. Si bien el estudiante a este nivel debería estar familiarizado con los conceptos, en el ámbito de un programa de cómputo un algoritmo es un texto (u otro medio) que describe sentencias que deben ejecutarse, un proceso por otro lado, es un "texto en acción", una entidad dinámica generada por la ejecución de un algoritmo. Un subproceso (llamado hilo a partir de aquí) es un proceso definido por un solo flujo de control (esto puede ser un programa individual, una computadora o una red) y en la mayoría de los casos es un componente de un proceso más general que lo ha generado.

Pueden existir múltiples hilos dentro de un proceso, ejecutándose simultáneamente y compartiendo recursos como la memoria, mientras que diferentes procesos no comparten este tipo de recursos.

La implementación de hilos difiere entre los sistemas operativos. Los sistemas con un solo procesador generalmente implementan hilos mediante un mecanismo de división de tiempo: la unidad central de procesamiento (CPU) cambia entre diferentes hilos de software. Este cambio de contexto generalmente ocurre suficientemente rápido como para que los usuarios perciban que los hilos o tareas se ejecutan en paralelo. En un sistema multiprocesador o multinúcleo, varios hilos pueden ejecutarse en paralelo, con cada procesador o núcleo ejecutando un subproceso separado simultáneamente.

El uso de hilos es un modelo generalizado de programación y ejecución que permite que existan múltiples subprocesos en el contexto de un proceso dado. Estos subprocesos comparten los recursos del proceso generador, pero pueden ejecutarse de forma independiente. Sin embargo, en un ambiente de programación donde se han generado varios hilos trabajando sobre elementos o controles en común del proceso padre, debe controlarse el acceso a estos recursos de manera secuencial, con el fin de evitar la generación de excepciones debido a que dichos elementos pueden estar siendo usados por otros hilos en un momento dado. Regularmente los distintos lenguajes de programación proveen mecanismos para evitar este tipo de problemas.

La programación concurrente tiene, entre otras, las siguientes ventajas:

* Aumento del rendimiento del programa: la ejecución paralela de un programa concurrente permite que la cantidad de tareas completadas en un tiempo determinado aumente proporcionalmente a la cantidad de procesadores del sistema.
* Permitir que una aplicación siga respondiendo a la interacción con el usuario. Si en un programa el subproceso de ejecución principal se bloquea en una tarea de ejecución prolongada, la aplicación completa puede aparecer como congelada. Al mover tales tareas de larga ejecución a un subproceso que se ejecuta simultáneamente con el proceso de ejecución principal, es posible que la aplicación siga respondiendo a las entradas del usuario mientras ejecuta tareas en segundo plano.
* Al usar subprocesos, una aplicación puede servir a múltiples clientes al mismo tiempo, utilizando menos recursos de los que necesitaría al usar múltiples copias de procesos de sí misma.

Muchos lenguajes de programación soportan hilos. Los lenguajes C, C++, Java, Python y .NET Framework, por mencionar algunos, exponen los subprocesos a los desarrolladores y permiten diferentes grados de usabilidad. Algunos lenguajes están diseñados primordialmente para la programación paralela, especialmente mediante la unidad de procesamiento de gráficos (GPU por sus siglas en inglés), sin requerir concurrencia o hilos (Ateji PX, CUDA).

***4. Material y equipo***



Computadora

***5. Desarrollo***

**I. Actividad 1**

**Implementación de hilos**

Realizar un programa que involucre un algoritmo de ejecución prolongada, de tal manera que el alumno pueda observar como su programa principal queda congelado mientras se ejecuta dicho algoritmo (programación secuencial). Una vez hecho esto, envíen la ejecución del algoritmo a un hilo (subproceso) y discutan el rendimiento del programa principal después de llevar a cabo tal acción (programación concurrente).

**II. Actividad 2**

**Evaluación de tiempos de ejecución**

Modifique el programa de la actividad anterior de tal manera que se ejecute al menos 10 veces el algoritmo de manera secuencial y concurrente. Evalué los tiempos de ejecución de las dos estrategias.

**III. Actividad 3**

**Sincronización de hilos**

Haga una modificación del algoritmo que se ha estado usando para que escriba en un archivo dado, envíe el algoritmo a los hilos necesarios para generar un error de acceso al archivo (debido a que otro hilo lo esté usando). Posteriormente implementen un mecanismo para sincronizar el acceso de los hilos al archivo.

***6. Bibliografía***

* BELL, Douglas y PARR, Mike. **C# para estudiantes**. México, Pearson, 2010
* BRINCH, Hansen. **The origin of concurrent programming**. New York, Springer, 2002.
* MCDOWELL, Laakmann. **Cracking the coding interview**. Palo Alto, CareerCup, 2015.
* RAYNAL, Michel. **Concurrent Programming: Algorithms, Principles, and Foundations**. New York, Springer, 2013.