CALDERÓN MORALES, MARCOS ANDRES 1060918
GIRÓN MÁRQUEZ, JOSÉ CARLOS 1064718
SIERRA ARAGÓN, YAZMINE ISABEL 1174916
SOLARES GARCÍA, ROBERTO ANTONIO 1173318
VELÁSQUEZ MORALES, IVANIA ALEJANDRA 1045718

Tarea Threads

Ventajas de threads sobre procesos

- Mejor tiempo de respuesta, puede ejecutarse el programa aunque esté bloqueado una parte.
- Los procesos son independientes, por lo tanto compartir información entre hilos se facilita.
- Simplifica el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.
- Es fácil la creación, cambio de contexto y gestión de hilos.
- Es menor el tiempo en terminar un hilo.
- Menos sobrecarga que fork.
- Un proceso es más costoso de crear, ya que se necesita crear una copia de toda la memoria de nuestro programa. Los hilos son más ligeros.

Ventajas de procesos sobre threads

- Los hilos se bloquean más fácilmente que un proceso.
- No todos los sistemas reconocen la existencia de hilos.
- Requiere de mayor espacio en el núcleo.
- Todas las llamadas al sistema las maneja el kernel, por lo tanto significa mayor costo.
- En cuanto a complejidad de implementación, en los hilos, al compartir la memoria y los recursos, es casi obligado el uso de mutex o semáforos, así que su programación suele ser más complicada.
- Los procesos de modo kernel son más seguros por la integridad que representan.

Threads en diferentes lenguajes

C

Ventajas:
- Rápidos para iniciar y terminar
- Comunicación rápida

Desventajas:
- Dificultad para encontrar errores

C++

Ventajas:
- Cuando se ejecuta una aplicación se crea un main thread en donde se pueden crear threads que ejecuten otras partes de código de la aplicación en paralelo.
- Son potentes y útiles

Desventajas:
- Los threads requieren de su propia pila en donde almacenan las variables locales, cada thread en un subproceso.
- Requieren de más procesamiento.
- Corrupción de datos

C#

Ventajas:
- Para todas y cada una de las aplicaciones de la interfaz de usuario, existe un único thread donde se ejecuta la aplicación. Pero si se ejecuta alguna tarea muy pesada, la interfaz de usuario podría quedarse congelada durante el tiempo que la tarea tarda en completarse. Esto se puede resolver ejecutando esa tarea larga en otro thread.
- En una operación de I/O la aplicación espera a que suceda alguna operación externa y luego continúa la ejecución, por ejemplo, leer el teclado. Con la ayuda de threads, se puede realizar alguna tarea con el CPU mientras se espera la respuesta del usuario.

Desventajas:
- Ejecución más lenta: en una máquina de un solo núcleo, la tarea que se realiza con un solo thread se realiza mucho más rápido que si la misma tarea la realizan varios threads debido al cambio de contexto.
- Mantenimiento: es difícil mantener el código escrito que realiza varias tareas utilizando varios threads.
- Debug: Es muy difícil depurar el código que se ejecuta con varios subprocesos.

Visual Basic

Ventajas:

- Evita que la interfaz de usuario se congele al ejecutar tareas pesadas en otros threads.
- Permiten la ejecución de tareas mientras se espera a que terminen I/O.

Desventajas:

- Debug: Es muy difícil depurar el código que se ejecuta con varios subprocesos.
- El lenguaje no permite pausar o eliminar threads directamente, se necesitan validar banderas definidas por el programador para hacerlo.

Python

Ventajas:

- Son adecuados para el manejo de I/O y la ejecución de tareas mientras se espera a que terminen operaciones de bloqueo (por ejemplo, esperar I/O, esperar resultados de una base de datos, etc).

Desventajas:

- Los threads de Python están restringidos a un modelo de ejecución que solo permite que se ejecute un thread en el intérprete en un momento dado. Por esta razón, los subprocesos de Python generalmente no deben usarse para tareas pesadas donde se intenta lograr el paralelismo en múltiples CPU.