MEDAC
Instituto Oficial de Formación Profesional

PROGRAMACIÓN DE SERVICIOS Y PROCESOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Introducción a la programación paralela o multihilo

04

ÍNDICE

OBJETIVOS

Aprender el concepto de hilo.

Comprender la diferencia entre hilos y procesos.

Conocer los estados por los que puede pasar un hilo en su ejecución.

Manejar las clases que aporta Java para la gestión de hilos.

/ 1. Introducción y contextualización práctica

En esta unidad, estudiaremos qué es la programación paralela, también conocida como multihilo.

Veremos qué son los hilos y en qué se diferencian de los procesos.

También conoceremos qué ventajas e inconvenientes presenta el uso de hilos en los diferentes programas que podemos crear.

Seguidamente, aprenderemos los recursos que pueden llegar a compartir los hilos y por los estados por los que estos podrán pasar a lo largo de su ejecución.

Por último, veremos qué clases nos proporciona Java para que podamos gestionar hilos de forma eficaz.

Todos estos conceptos los pondremos en práctica a la vez que se vayan exponiendo, así que los aprenderás con facilidad.

Fig. 1. Estudiando hilos.

Escucha el siguiente audio, en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad:

Audio Intro. "Invernadero"

https://bit.ly/2ZUVQ2R

/ 2. Hilos de ejecución en un proceso

Seguramente, no será la primera vez que navegas por Internet con tu navegador favorito mientras escuchas música y, al mismo tiempo, te estás descargando un fichero a tu disco duro. Todas estas tareas las está realizando el propio navegador simultáneamente, es decir, estás escuchando música y el fichero se está descargando al mismo tiempo.

Esto se debe a que los navegadores web están utilizando la programación paralela o multihilo. Este tipo de programación es un tipo de programación concurrente capaz de ejecutar al mismo tiempo varias tareas o hilos.

Un hilo, también conocido como hebra, no es más que un trozo de código que se ejecuta dentro de un proceso, pero que tiene la enorme ventaja de que puede ser ejecutado en paralelo con otros hilos.

Que los hilos se ejecuten dentro de un proceso significa que el proceso podrá crearlos y lanzarlos, pero estos solo podrán utilizar los recursos de los que el propio proceso disponga. Con esto hay que tener mucho cuidado, ya que los diferentes hilos de un proceso pueden necesitar acceder a los mismos recursos, por lo que se puede producir una situación bastante indeseable y ocasionar lo que se conoce como «inconsistencias» en nuestros programas.

Los procesos activos seguirán en ejecución hasta que todos y cada uno de los hilos que han lanzado terminen de ejecutarse. En ese preciso momento, el proceso podrá ser destruido por el sistema operativo y todos los recursos de los que disponía serán liberados.

Cuando ejecutamos un programa, se crean también tanto un proceso como un hilo primario.

Sobre los hilos, debemos tener en cuenta que:

- Los hilos no pueden existir de forma independiente a un proceso.
- Los hilos no se podrán ejecutar por sí solos.
- Dentro de un mismo proceso podremos tener tantos hilos ejecutándose como necesitemos.

Si tenemos un único hilo, tendremos algo muy similar a un programa secuencial normal y corriente. La verdadera diferencia reside en ejecutar varios hilos de forma concurrente.

Audio 1. "Diferencia entre proceso e hilo"
https://bit.ly/3jubfyy

/ 3. Ventajas y desventajas del uso de hilos

Cuando hablamos de hilos, podemos decir que sus ventajas aparecen una vez tenemos más de uno. Es decir, cuando tenemos programación multihilo. En este caso, dentro de un mismo proceso, tendremos múltiples hilos en ejecución, los cuales estarán realizando actividades totalmente distintas, pudiendo o no cooperar entre ellos.

Son muchas las ventajas que presentan los hilos en comparación con los procesos. Algunas de ellas son:

- **Compartir recursos:** Dentro de un proceso, todos sus hilos van a compartir tanto la memoria como los recursos de los que disponga. Gracias a esto, todas las operaciones de los hilos serán mucho más rápidas.
- **Uso más eficiente y ahorro de memoria:** Al compartir la misma zona de memoria, la operación de creación de nuevos hilos en el proceso no va a suponer una reserva adicional de esta.

- **Capacidad de respuesta:** El hecho de tener varios hilos permitirá al proceso atender otras peticiones que le envíe el usuario a través de los otros hilos.

- **Paralelismo real:** En los sistemas en los que el microprocesador es multinúcleo, los hilos se podrán ejecutar en un núcleo diferente cada uno, lo cual permite usar el procesador de forma paralela, haciendo que se ejecuten varias instrucciones al mismo tiempo.

Pero, como todo, los hilos también presentan una serie de desventajas:

- No todos los lenguajes de programación soportan la programación multihilo, aunque, actualmente, casi todos si lo hacen.

- Es tarea del programador controlar todos los problemas que estén relacionados con la comunicación y sincronización de los hilos. Los problemas que se darán cuando estos comparten recursos son inanición, bloqueo activo, acceso a los recursos críticos, zonas de exclusión mutua, condiciones de carrera o errores de inconsistencia en la memoria compartida, entre otros.

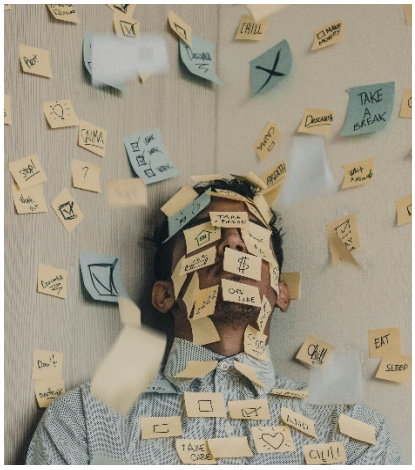

Fig. 2. Analogía de varios hilos ejecutándose.

/ 4. Caso práctico 1: "¿Es mejor utilizar hilos o procesos?"

Planteamiento: Pilar y José han comenzado a estudiar los conceptos sobre programación multihilo e hilos y, aunque parecen conceptos bastante complicados, no se dan por vencidos.

Una de las cosas que han notado rápidamente ha sido que los conceptos de hilos y procesos son bastante parecidos. «¿No crees que es lo mismo que vimos en la unidad de procesos?», le comenta Pilar a José, a lo que él le responde que tiene razón, los conceptos parecen los mismos.

Nudo: ¿Crees que hilos y procesos son lo mismo? ¿Podríamos dejar a un lado los hilos y trabajar únicamente con procesos?

Desenlace: Cuando se estudia la programación concurrente, tanto procesos como hilos, al principio es muy común pensar que estos conceptos son iguales, pero nada más lejos de la realidad.

Los hilos y los procesos son conceptos totalmente diferentes, aunque es cierto que están fuertemente ligados entre ellos.

Quizá, la diferencia más grande entre estos dos conceptos sea que un proceso se encargará de crear los hilos y que estos se ejecutarán únicamente en el espacio de memoria del proceso que los ha creado. Por lo tanto, si hay más de un proceso con varios hilos creados y ejecutándose, estos nunca «verán» los hilos del otro proceso. En cambio, cuando un proceso crea otro proceso hijo, ocurre totalmente lo contrario, ya que este podrá interactuar con otros procesos que existan.

Por todo esto, podemos concluir que el usar procesos o hilos para resolver un problema dependerá del mismo: si es un único problema con varias partes identificables, podremos usar hilos; pero, si son varios problemas diferentes dentro del mismo, podremos usar procesos.

/ 5. Recursos compartidos por los hilos

Todos los hilos están compuestos por los siguientes elementos:

- Un identificador único, mediante el cual cada hilo podrá ser identificado de forma rápida.
- Un contador de programa, mediante el cual el hilo podrá ejecutar su código de forma independiente.
- Un conjunto de registros asociados, con los que el hilo podrá realizar todas las operaciones aritmético-lógicas que necesite de forma independiente.
- Una pila propia, con la que el hilo podrá ejecutar las llamadas a funciones que necesite de forma independiente.

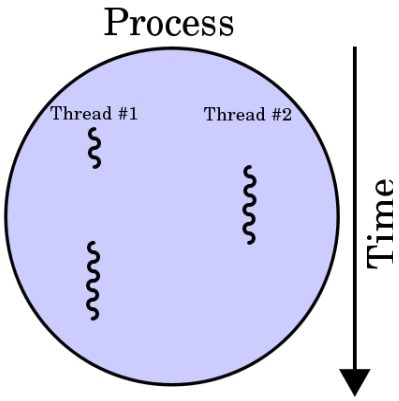

Fig. 3. Varios hilos en un proceso.

Los hilos también podrán compartir una serie de recursos con otros como, por ejemplo:

- El código a ejecutar.
- Variables globales que se encontrarán en la zona crítica.
- Recursos del sistema operativo, como ficheros, sockets, bases de datos, etc.

Que los hilos puedan compartir recursos es muy peligroso, ¿qué ocurriría si dos o más hilos necesitan acceder a una misma variable global para realizar un cierto cálculo, pudiendo cambiar el valor de dicha variable? Si esto ocurre y no se controla, podría suceder que el primer hilo que acceda a la variable cambie su valor y que, por consiguiente, los demás hilos, cuando accedan a ella, no accedan al valor anterior, sino al cambiado, lo cual podrá ocasionar efectos imposibles de predecir en el comportamiento de nuestro programa.

Todo esto se puede solucionar utilizando lo que conocemos como un esquema de bloqueo y sincronización entre varios hilos.

Estos sistemas de bloqueo y sincronización resolverán ese problema, pero su implementación no es nada sencilla y complicarán bastante nuestros programas.

Más adelante, veremos más detenidamente algunos de estos mecanismos.

Vídeo 1. "Ejemplo de varios hilos"

https://bit.ly/2BjZ8Tw

/ 6. Estados de un hilo

Los hilos, al igual que los procesos, tienen un ciclo de vida determinado por una serie de estados. Estos son iguales tanto para los hilos que crea el usuario como para los que crea el sistema, conocidos como hilos demonio o de sistema.

El comportamiento de cada hilo dependerá del estado en el que se encuentre, de forma que va a definir la operación que está realizando. Más adelante, veremos cómo podemos obtener el estado en el que se encuentra un hilo de forma sencilla.

Los estados por los que puede pasar un hilo nos resultan familiares a los vistos en el tema 2 con los procesos, ya que son los mismos. No obstante, cada estado tiene unas peculiaridades:

- **Nuevo:** En este estado, el hilo ya ha sido creado y está listo para ejecutarse, aunque no haya sido elegido para empezar a ejecutarse.
- **Listo o ejecutable:** El hilo entrará en este estado cuando indiquemos, con su método correspondiente, que está listo para poder ejecutarse.
- **En ejecución:** En este estado, el hilo está listo para ejecutarse y el sistema operativo lo ha seleccionado para que se ejecute.
- **Bloqueado:** El hilo entrará en este estado cuando necesite algunos recursos de entrada/salida que debe proporcionarle el usuario o el propio sistema operativo. En este estado, no se le va a asignar tiempo de CPU al hilo, aunque estará listo para volver a ser usado.
- **Finalizado:** Cuando el hilo termina su ejecución, pasa a este estado, en espera de que el sistema operativo lo destruya y libere sus recursos.

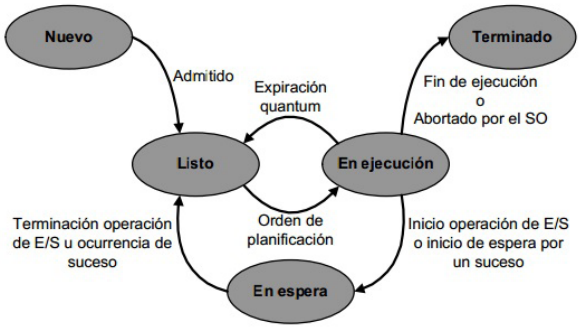

Fig. 4. Diagrama de estados de un hilo.

/ 7. Clases para hilos en Java

El lenguaje de programación Java nos ofrece varias clases para poder realizar programas que usen la programación paralela o multihilo. Dentro del paquete java.lang encontraremos todas estas utilidades:

- **Clase Thread:** Con esta clase, podremos crear hilos totalmente funcionales a los que podremos asignar el código que queramos para que lo ejecuten. Las clases que queramos que sean hilos deberán heredar de esta.
- **Interfaz Runnable:** Con esta interfaz, podremos añadir la funcionalidad de hilo a cualquier otra clase por el mero hecho de implementarla.

- **Clase ThreadDeath:** Con esta clase, podremos manejar y notificar errores en el uso de las hebras. Hereda de la clase Error.
- **Clase ThreadGroup:** Con esta manejaremos un grupo de hilos de forma conjunta, haciendo que se ejecuten de una forma bastante más eficiente.

Dentro de la clase Thread, podremos encontrar, entre muchos otros, los siguientes métodos:

- **new():** Este método creará un hilo.
- **start():** Indica que el hilo está listo para ejecutarse.
- **run():** Ejecuta el hilo, que empezará a ejecutarse de forma paralela.
- **sleep():** Hace que el hilo se bloquee una cantidad de tiempo dada en milisegundos.
- **wait():** Hace que el hilo espere la ejecución de otra tarea para volver a ejecutarse.
- **getState():** Este método nos devolverá el estado en el que se encuentra actualmente el hilo.

Las clases que más vamos a utilizar con este propósito serán la clase Thread y la interfaz Runnable.

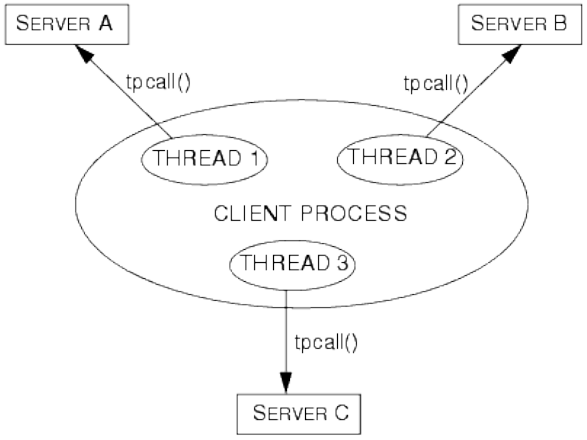

Fig. 5. Ejemplo de uso de hilos.

Vídeo 2. "Documentación de las clases Thread y Runnable"
https://bit.ly/3hp2x2U

/ 8. Caso práctico 2: "¿Qué lenguajes soportan la programación multihilo?"

Planteamiento: Una vez Pilar y José comienzan a entender cómo funciona la programación multihilo y todas las ventajas que esta nos ofrece, la empiezan a ver con otros ojos.

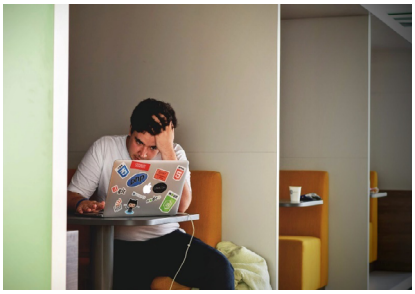

Fig. 6. Estudiando programación multihilo.

«Quizá no sea tan complicada como la pintan», le dice Pilar a José. Este, pensativo, le da la razón, pero se pregunta si esto de la programación multihilo es solo cosa de Java y si los demás lenguajes no la van a soportar.

Nudo: ¿Qué piensas al respecto? ¿Crees que los demás lenguajes de programación soportan la programación multihilo o es solo cosa de Java?

Desenlace: La programación multihilo es algo muy extendido hoy en día. De hecho, no tardó mucho en desarrollarse cuando empezaron a surgir los lenguajes de programación, ya que la necesidad de ejecutar varias tareas al mismo tiempo fue decisiva.

Según esto, la programación multihilo no es solo cosa de Java. Cualquier lenguaje de programación potente de hoy en día la va a soportar, algunos ejemplos son: C, C++, C#, Python, Kotlin, Swift y un larguísimo etcétera.

/ 9. Ejemplo de hilos: caja

Un ejemplo que quizá nos haga ver de una forma mucho más clara las ventajas de la programación paralela o multihilo es el siguiente.

Vamos a simular el proceso de una cola en un supermercado y el cobro por parte de los empleados de caja. Para esto, imaginemos que una serie de clientes van a pagar con sus carros llenos de productos.

Si no utilizásemos programación multihilo, únicamente tendríamos una caja abierta, en donde se iría pasando, uno a uno, todos los artículos de todos los clientes. Este proceso sería muy lento, ya que, para dar servicio a un determinado cliente, debería haberse terminado el servicio con los clientes anteriores.

Podríamos decir que el esquema de la siguiente figura se aproxima a la ejecución del programa.

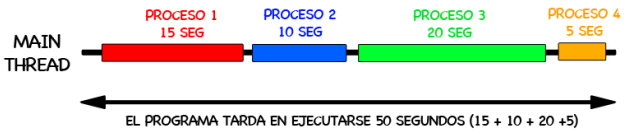

Fig. 7. Ejecución del ejemplo sin multihilo.

Vamos a aplicar ahora un poco de multihilo para hacer más eficiente y rápido nuestro programa.

Para ello, esta vez colocaremos 4 cajas con 4 empleados, en lugar de solamente 1. De esta forma, habrá 4 cajas disponibles por las que los clientes podrán ir pasando para pagar sus productos. Cada empleado de caja se va a comportar exactamente igual que en el anterior caso: pasará, uno a uno, todos los productos de cada cliente.

No es difícil deducir en qué modelo los clientes van a tardar menos en ser atendidos. Por lo tanto, llevando el modelo a la programación multihilo, concluimos que esta provocará que el proceso se ejecute mucho más rápido.

En la siguiente figura, podemos ver un esquema aproximado de cómo sería el comportamiento del programa.

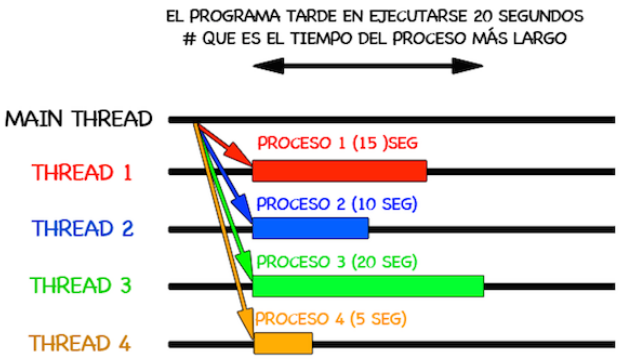

Fig. 8. Ejecución del ejemplo utilizando multihilo.

/ 10. Resumen y resolución del caso práctico de la unidad

A lo largo de esta unidad, hemos visto qué es la programación paralela o multihilo.

Hemos expuesto qué ventajas e inconvenientes implica usar este tipo de programación.

También hemos aprendido que los hilos pueden llegar a compartir recursos. Hemos estudiado cuáles son estos y los peligros que esto conlleva a la hora de programarlos.

Seguidamente, hemos repasado los estados por los que podrá pasar un hilo en toda su vida útil (muy similares a los de los procesos), desde que es creado y lanzado hasta que termina su ejecución.

Por último, hemos profundizado en las clases que nos proporciona el lenguaje de programación Java para que podamos gestionar hilos.

Resolución del Caso práctico de la unidad

El problema del invernadero es un ejemplo fantástico para el uso de programación multihilo.

Nuestros amigos tienen que poder controlar muchos elementos como, por ejemplo, la temperatura, la humedad o el riego, que deberá ser lanzado a ciertas horas y cortado a otras.

Todas estas tareas serían imposibles de realizar mediante la programación convencional a la que estamos acostumbrados, ya que no podríamos estar tomando muestras de la humedad y la temperatura al mismo tiempo, por ejemplo.

Lo óptimo sería utilizar varios hilos, uno para cada tarea y, como tenemos la ventaja de que se ejecutarán simultáneamente, no tendremos problemas a la hora de realizar todas y cada una de las tareas de invernadero.

/ 11. Bibliografía

Gómez, O. (2016). Programación de Servicios y Procesos Versión 2.1. Recuperado de: https://github.com/OscarMaestre/ServiciosProcesos/blob/master/_build/latex/Servicios Procesos.pdf

Hilo (informática), en Wikipedia, la enciclopedia libre. Recuperado de: https://es.wikipedia.org/wiki/Hilo_(inform%C3%A1tica)

Moya, R. (23 de mayo de 2014). [Entrada de Blog]. Multitarea e Hilos en Java con ejemplos (Thread & Runnable). Jarroba. Recuperado de: https://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/

Sánchez, J. M. y Campos, A. S. (2014). Programación de servicios y procesos. Madrid: Alianza Editorial.

Thread (computing), en Wikipedia, la enciclopedia libre. Recuperado de: https://en.wikipedia.org/wiki/Thread_(computing)