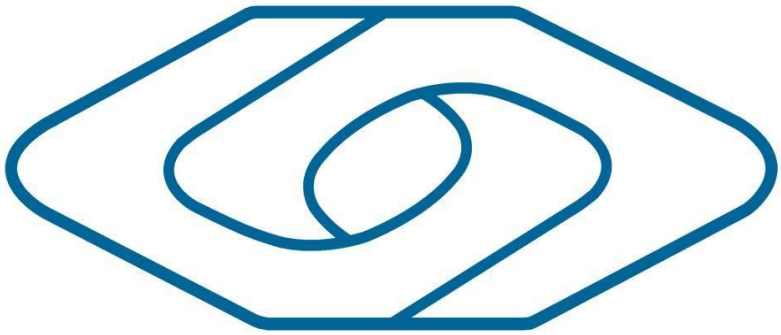
INSTITUTO POLITÉCNICO NACIONAL



ESCOM

ING. EN SISTEMAS COMPUTACIONALES

APLICACIONES PARA COMUNICACIONES EN RED

“TAREA 1: HILOS”

PROFESOR: RANGEL GONZALEZ JOSUE

INTEGRANTES:

DOMINGUEZ MARTÍNEZ BRAULIO SEBASTIÁN

GONZALEZ RODRIGUEZ LUIS GUILLERMO

GRUPO: 3CV16

INDICE

INTRODUCCION

Al describir cómo funcionan las computadoras a alguien nuevo en las PC, a menudo es más fácil sacar la vieja noción de que un programa es una colección muy grande de instrucciones que se ejecutan de principio a fin. Nuestra noción de un programa puede incluir ciertas excentricidades, como bucles y saltos, que hacen que un programa se asemeje más a un juego de toboganes y escaleras que a un rollo de piano. Si las instrucciones de programación fueran cuadrados en un tablero de juego, podemos ver que nuestro programa tiene lugares en los que nos estancamos, cuadrados que cruzamos una y otra vez y puntos que no cruzamos en absoluto. Pero tenemos una forma de entrar en nuestro programa, independientemente de sus giros y saltos, y una salida.

No hace muchos años, las instrucciones individuales eran la forma en que enviábamos el trabajo a las computadoras. Desde entonces, las computadoras se han vuelto cada vez más poderosas y más eficientes para realizar el trabajo que hace posible la ejecución de nuestros programas. Las computadoras de hoy pueden hacer muchas cosas a la vez (o hacernos creer que sí). Cuando empaquetamos nuestro trabajo de acuerdo con la noción tradicional y serial de un programa, le estamos pidiendo a la computadora que lo ejecute de manera similar al rendimiento humilde de una computadora de ayer. Si todos nuestros programas se ejecutan así, es muy probable que no estemos usando nuestra computadora al máximo de sus capacidades.

Una de esas capacidades es la capacidad de un sistema informático para realizar múltiples tareas. Hoy en día, con frecuencia es útil ver nuestro programa (nuestra gran tarea) como una colección de subtareas. Por ejemplo, si nuestro programa es un sistema de navegación marina, podríamos ejecutar tareas separadas para realizar cada sondeo y mantener otras tareas que calculan la profundidad relativa, correlacionan las coordenadas con las mediciones de profundidad y muestran cartas en una pantalla. Si podemos hacer que la computadora ejecute algunas de estas subtareas al mismo tiempo, sin cambios en los resultados de nuestro programa, nuestra tarea general continuará obteniendo todo el procesamiento que necesita, pero se completará en un período de tiempo más corto. En algunos sistemas, la ejecución de subtareas se intercalará en un solo procesador; en otros, pueden funcionar en paralelo. De cualquier manera, veremos un aumento en el rendimiento.

Hasta ahora, cuando dividíamos nuestro programa en múltiples tareas, solo teníamos una forma de entregarlas al procesador: los procesos. Específicamente, comenzamos a diseñar programas en los que los procesos principales bifurcaban a los procesos secundarios para realizar subtareas. En este modelo, cada subtarea debe existir dentro de su propio proceso. Ahora, se nos ha brindado una alternativa que es aún más eficiente y proporciona un rendimiento aún mejor para nuestro programa general: subprocesos. En el modelo de subprocesos, existen múltiples subtareas como flujos de control individuales dentro del mismo proceso.

El modelo de hilos toma un proceso y lo divide en dos partes:

Uno contiene los recursos utilizados en todo el programa (la información de todo el proceso), como las instrucciones del programa y los datos globales. Esta parte todavía se conoce como el proceso.

El otro contiene información relacionada con el estado de ejecución, como un contador de programa y una pila. Esta parte se conoce como un hilo.

DESARROLLO

1. ¿Qué son los hilos?

Un hilo es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea.

Un hilo o también conocido como proceso ligero es una unidad básica de ejecución, que cuenta con su propio contador de programa registros de la unidad de procesamiento central (pila).

Los hilos dentro de una misma aplicación comparten; código y datos recursos del sistema operativo, como lo son ficheros, objetos de entrada y salida, entre otros.

La creación de un nuevo hilo es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso compartan los recursos hace que cualquiera de estos hilos pueda modificar dichos recursos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente.

Lo que es propio de cada hilo es el contador de programa, la pila de ejecución y el estado de la CPU (incluyendo el valor de los registros).

El proceso sigue en ejecución mientras al menos uno de sus hilos de ejecución siga activo. Cuando el proceso finaliza, todos sus hilos de ejecución también han terminado. Asimismo, en el momento en el que todos los hilos de ejecución finalizan, el proceso no existe más y todos sus recursos son liberados.

Algunos lenguajes de programación tienen características de diseño expresamente creadas para permitir a los programadores lidiar con hilos de ejecución (como Java o Delphi). Otros (la mayoría) desconocen la existencia de hilos de ejecución y éstos deben ser creados mediante llamadas de biblioteca especiales que dependen del sistema operativo en el que estos lenguajes están siendo utilizados (como es el caso del C y del C++).

Un ejemplo de la utilización de hilos es tener un hilo atento a la interfaz gráfica (iconos, botones, ventanas), mientras otro hilo hace una larga operación internamente. De esta manera el programa responde de manera más ágil a la interacción con el usuario. También pueden ser utilizados por una aplicación servidora para dar servicio a múltiples clientes.

2. Diferencias entre hilos y procesos

Un hilo es una línea de ejecución de un proceso. Todo proceso parte inicialmente con un único hilo principal, aunque el sistema operativo ofrece llamadas al sistema que permiten al programador crear y destruir hilos. Por tanto, un proceso está compuesto por uno o más hilos.

Los estados de un hilo son iguales a los de un proceso, por tanto, un hilo puede estar en estado preparado, bloqueado o activo en un cierto instante de tiempo. La conmutación entre hilos de un proceso es menos costosa que la conmutación de procesos, por tanto, el planificador tiende a conmutar entre hilos de un proceso siempre que el proceso en su conjunto no haya agotado el tiempo máximo de asignación del procesador.

Los hilos nos permiten aprovechar la existencia de más de un procesador en el sistema, puesto que podemos asignar un hilo a cada uno de los procesadores que haya disponibles. Si hay más de un procesador, dos hilos de un mismo proceso pueden estar en estado activo simultáneamente. Por tanto, la programación con hilos nos permite sacar partido de las arquitecturas de multiprocesador que predominan en la actualidad.

Cuando se crea un hilo, el programador indica qué código ejecuta. Los hilos de un mismo proceso comparten el mismo espacio de memoria, por tanto, dos hilos del mismo proceso pueden compartir estructuras de datos, variables, código, archivos abiertos, etc. Al acceder a los recursos del sistema de manera compartida, las tareas realizan un consumo de recursos inferior.

Con multiprocesamiento podemos ejecutar diferentes procesos a la vez. Si tenemos un solo proceso podemos desdoblarlo en múltiples hilos. Para aumentar su eficiencia, un programa en ejecución debe crear tantos hilos como el doble del número de procesadores de los que disponga el sistema.

Las principales ventajas del uso de hilos son:

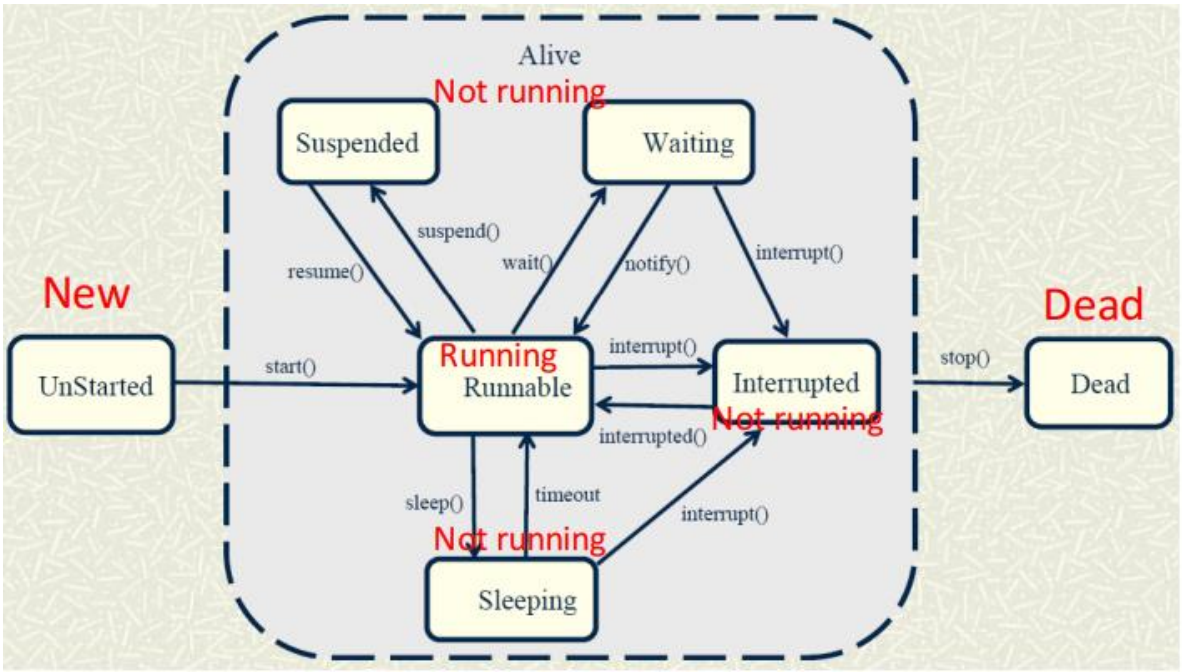
- Menor penalización en cuanto a conmutación. Ya que sólo se produce un salto en la ejecución de código del proceso, no interviene el planificador de procesos.
- Dos hilos de un mismo proceso comparten memoria aprovechando mejor los recursos y eliminando los mecanismos de comunicación necesarios para soluciones implementadas en varios procesos.
- Con un solo procesador es posible que una línea de proceso haga uso de una llamada al sistema que haga que el proceso pase a estado bloqueado. En este caso se puede llamar a otro hilo del mismo proceso en lugar de conmutar a otro proceso externo, ahorrando así tiempo de conmutación asociado a la ejecución del planificador de procesos.

3. Ciclo de vida de hilos

El comportamiento de un hilo depende del estado en que se encuentre, este estado define su modo de operación actual. Los estados en los que puede estar un hilo son los siguientes:

- New: Un hilo se encuentra en estado new la primera vez que se crea y hasta que el método start es llamado. Los hilos en estado new ya han sido inicializados y están listos para empezar a trabajar, pero aun no han sido notificados para que empiecen a realizar su trabajo.
- Running: Cuando se llama al método start de un hilo nuevo, el método run es invocado y el hilo entra en el estado de running. Este estado podría llamarse "running" porque la ejecución del método run significa que el hilo está corriendo.
- Not running: El estado not running se aplica a todos los hilos que están parados por alguna razón. Cuando un hilo esta en este estado, esta listo para ser usado y es capaz de volver al estado running en un momento dado. Los hilos pueden pasar al estado not running a través de varias vías.
    - El método suspend()
    - El método sleep()
    - El método wait()
    - El método interrupt()
- Dead: Un hilo entra en estado dead cuando ya no es un objeto necesario. Los hilos en estado dead no pueden ser resucitados y ejecutados de nuevo. Un hijo puede entrar en estado dead a través de dos vías:
    - El método run termina su ejecución.
    - El método stop es llamado.

  La primera es una muerte natural, la segunda es una muerte causada.



4. API para programar hilos en C

La función que para crear un nuevo hilo de ejecución es pthread_create() y admite 4 parámetros:

- pthread_t * es un puntero a un identificador de thread. La función nos devolverá este valor relleno, de forma que luego podamos referenciar al hilo para trabajar con él.
- pthread_attr_t * son los atributos de creación del hilo. Hay varios atributos posibles, como por ejemplo la prioridad. Un hilo de mayor prioridad se ejecutará con preferencia (tendrá más rodajas de tiempo) que otros hilos de menor prioridad. Se puede pasar NULL, con lo que el hilo se creará con sus atributos por defecto y para nuestro ejemplo es suficiente. Si queremos un programa que cree y destruya hilos continuamente, no vale NULL, ya que con esta opción dejaremos memoria sin liberar cada vez que termine un hilo.
- void *(*)(void *). Aunque asuste, no es más que el tipo de una función que admite un puntero void * y que devuelve void *. Eso quiere decir que a este parámetro le podemos pasar el nombre de una función que cumpla lo que acabamos de decir. Esta función es la que se ejecutará como un hilo aparte. El hilo terminará cuando la función termine o cuando llame a la función pthread_exit() (o que alguien lo mate desde otra parte del código). Es bastante habitual hacer que esta función se meta en un bucle infinito y quede suspendida en un semáforo o a la espera de una señal para hacer lo que tenga que hacer y volver a quedar dormida.
- void * es el parámetro que se le pasará a la función anterior cuando se ejecute en el hilo aparte. De esta manera nuestro programa principal puede pasar un único parámetro (que puede ser cualquier cosa, como una estructura compleja) a la función que se ejecutará en el hilo. La función del hilo sólo tendrá que hacer el "cast" adecuado.

```
void *funcionDelHilo (void *parametro)
{
    Estructura *miEstructura = (Estructura *)parametro;
    ...
}
```

La función pthread_create() devuelve 0 si todo ha ido bien. Un valor distinto de 0 si ha habido algún problema y no se ha creado el thread.

```
void *funcionDelThread (void *);
...
pthread_t idHilo;
...
pthread_create (&idHilo, NULL, funcionDelThread, NULL);
```

Podemos hacer que un hilo espere por otro. Para ello tenemos la función pthread_join().

Podemos hacer que dos o más hilos accedan sincronizadamente a un recurso común. Para ello tenemos las funciones pthread_mutex_lock() y similares.

El sistema operativo pone en marcha uno de los threads y no le quita el control hasta que él lo diga. Los demás threads quedan parados hasta que el primero "ceda el control". Para ceder el control suele haber funciones estilo yield(), thr_yield(), pthread_delay_pn(), etc. También se cede el control si se hacen llamadas que dejen dormido al thread en espera de algo, como wait() o sleep().

CONCLUSIONES

1. La programación multihilo sin duda tiene mucho campo de aplicación, desde los sistemas operativos hasta en la tecnología que usamos cotidianamente como los celulares, cajeros etc.

En la elaboración de este trabajo obtuvimos conceptos que nos darán la fluidez para desarrollar aplicaciones utilizando multihilo, con la finalidad de poder realizar más de una tarea a la vez y administrando correctamente los recursos del ordenador.

2. La utilidad de la programación multihilos resulta evidente. Por ejemplo, un navegador Web puede descargar un archivo de un sitio, y acceder a otro sitio al mismo tiempo. Si el navegador puede realizar simultáneamente dos tareas, no tendrá que esperar hasta que el archivo haya terminado de descargarse para poder navegar a otro sitio.

En conclusión, la programación multihilo está presente es la mayor parte de las aplicaciones informáticas que usamos cotidianamente sin importar el lenguaje en que se desenvuelvan, el concepto de Thread o Hilo seguirá siendo el mismo.

REFERENCIAS

http://aisii.azc.uam.mx/areyes/archivos/licenciatura/sd/U2/ConceptoHilos.pdf

https://webprogramacion.com/procesos-e-hilos/

https://es.wikipedia.org/wiki/Hilo_(inform%C3%A1tica)

https://www.fing.edu.uy/tecnoinf/mvd/cursos/so/material/teo/so05-hilos.pdf

https://www.cartagena99.com/recursos/alumnos/apuntes/Tema2.L3-Hilos%20y%20Procesos.pdf

https://sites.google.com/site/carlosraulsan2987/home/sistemas-operativos/unidad-2-y-3/procesos-e-hilos

https://es.slideshare.net/EmmanuelGarcaFortuna/procesos-e-hilos-en-los-sistemas-operativos

https://1984.lsi.us.es/wiki-ssoo/index.php/Hilos

https://www.chuidiang.org/clinux/procesos/procesoshilos.php#:~:text=programaci%C3%B3n%20de%20hilos-,Procesos%20y%20Threads%20(hilos%20de%20ejecuci%C3%B3n),de%20ejecuci%C3%B3n%20(un%20thread).

https://www.it.uc3m.es/pbasanta/asng/course_notes/c_threads_functions_es.html

https://www.um.es/earlyadopters/actividades/a3/PCD_Activity3_Session1.pdf