# Teoría 4 - Hilos

## Hilos

También llamado hebra, proceso ligero o Thread (en ingles) es una unidad básica de ejecución. Todos los procesos parten de un único hilo principal y el SO ofrece system calls para que el desarrollado cree y destruya hilos. Cada hilo tiene elemento propios como contador de programa (PC), registros de CPU, pila (stack) y estado; pero los hilo de una misma aplicación comparten a su vez otro elementos, como espacio de direcciones, variables globales, archivos abiertos, procesos hijos, etc.

Un *proceso* puede contener uno o varios hilos compartiendo memoria y recursos, por lo que se podría decir que un proceso es una aplicación y los hilos son actividades concurrentes dentro de esa aplicación.

### Multithreading

La ventaja de usar hilo por sobre procesos es que, al comparitr recursos, los hilos son más fáciles de crear, finalizar e intercambiar. Además, nos permite modularizar un programa, permitiendo que cada hilo realice una tarea en particular. Todos los hilos tienen los mismos privilegios y pueden acceder al estado de otros hilos.
Hay acciones que afectan a todos los hilos de un proceso, como por ejemplo un cambio de contexto o el suspender el mismo, lo cual suspende todos los hilos y estos terminan.

### Estados de un hilo

Los estados son:

- En ejecución
- Listo
- Bloqueado

Las operaciones relacionadas a estos estados son:

- Creación: al crear un proceso, también se crea un hilo principal de el mismo, que puede crear otros dentro del mismo proceso dándo punteros al mismo código y argumentos. Este hilo se coploca en la cola de Listo.
- Bloqueo: un hilo se bloquea cuando necesita esperar un evento. El CPU puede pasar a ejecutar otro hilo en estado Listo del mismo proceso o de otro.
- Desbloqueo: cuando se cumple el evento que bloquea el hilo, pasa a Listo.
- Planificación y expulsión: corresponden a la transición entre *Listo* y *En Ejecución*, y se llevan a cabo de acuerdo a las políticas de planificación.

- Finalización: cuando termina el hilo, se libren sus registros y pila.

## Implementación ULT

En los hilos a nivel de usuario (User Level Thread), la gestión de los hilos (creacion, finalización, intercambio de mensajes, etc) la realiza la aplicación. Estas operaciones se realizan en el espacio de usuario de un mismo proceso. Mientras tanto, el SO planifica el proceso como una unidad con un único estado, indistinto a la existencia de hilos. En POSIX, estos son los llamados pthreads.
Ventajas:

- Más rápidos, ya que no requieren llamadas al sistema.
- No es necesario tener privilegios de kernel para intercambiar hilos.
- Se puede seleccionar una planificación específica.
- Pueden ejecutarse en cualquier SO.

Desventajas:

- Una llamada al sistema bloquea no solo el hilo sino también al resto de hilos del proceso.
- No se aprovechan las ventajas de multiprocesamiento ya que el procesador ve a todos los hilos del proceso como uno solo.

## Implementación KLT

En los hilos a nivel de kernel (Kernel Level Thread), la gestión de los hilos la realiza el kernel, mientras que la aplicación solo puede hacer uso de una API para la gestión.
Ventajas:

- El kernel puede planificar hilos de múltiples procesos en múltiples procesadores.
- Si se bloquea un hilo, puede planificarse otro del mismo proceso (no se bloquea todo el proceso padre).
- Las propias funciones del kernel pueden ser multihilo.

Desventajas:

- El paso de control de un hilo a otro se realiza accediendo al kernel.

## Combinaciones de ULT y KLT

Existen aproximaciones combinadas de ULT y KLT, donde

- La creación, sincronización y planificación se da en modo usuario.
- Se puede planificar múltiples hilos de un mismo proceso en múltiples procesadores.
- El bloqueo de un hilo no supone necesariamente el bloqueo de todo el proceso.