




## **Threads (Hilos):**

- Es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.
- Un solo proceso puede tener varios hilos asociados.
- Comparten Address Space.
- Varios hilos comparten un solo código y espacio de Heap.
- Cada hilo tiene su propio program counter y espacio de Stack.
- El PCB(process control block) pasa a ser TCB(thread control block).

 **Concurrencia:** Un sistema que ejecuta todas las tareas por un corto periodo de tiempo.

**Vs**

 **Paralelismo:** Un sistema que ejecuta varias tareas de forma simultánea a lo largo del tiempo.

 **NOTA:** Una aplicación que implemente paralelismo siempre tendrá concurrencia, sin embargo esto no ocurre al contrario.

## **Cambio de contexto entre hilos (de un mismo proceso):**

- No es necesario cambiar el espacio de código y de Heap, este es compartido entre hilos.
- Se cambia el espacio de stack y se actualiza el PC.
- Es más eficiente que el cambio de contexto en procesos.

## **Race Condition:**



Se da cuando 2 hilos intentan acceder a la misma posición de memoria de forma simultánea. Puede generar inconsistencias en las variables accedidas.


## **Región crítica:**

Porción de código que accede una variable compartida, se pueden implementar mecanismos de exclusión mutua para evitar que se acceda de forma simultánea.

## **Locks:**

Elementos de programación que permiten manejar secciones críticas, un lock puede estar:

-  **Disponible:** puede ser adquirido por un hilo para acceder a la región crítica.
-  **Adquirido:** está siendo usado por un hilo para acceder a la región crítica.

 **NOTA:** Se pueden usar varios locks para diferentes secciones críticas, esto incrementa la concurrencia.

## **Implementación de un lock:**

Es necesario soporte de hardware y/o sistema operativo, un lock debe de cumplir con:

- Exclusión mutua
- Equidad (accesible por todos los hilos, no inanición)
- Desempeño

## **Instrucciones atómicas:**

- No interrumpibles por el scheduler o sistema operativo
- Permite la implementación de locks mediante hardware

## **Spin-Lock:**

Lock que implementa una espera activa(ciclo while infinito), puede fallar comúnmente la condición de equidad.



## **Cómo evitar el gasto de CPU en los locks:**

Se busca eliminar el tiempo de espera activa, pues en este tiempo la cpu no ejecuta nada.

- yield(): Este llamado devuelve el control a la cpu.
- park(): el hilo entra en modo de espera o se “Duerme”.
- unpark(Thread ID): “Despierta” o saca de espera un hilo en particular.

## **Semáforos:**

Objetos de programación con un valor entero, como un contador, que pueden cumplir funciones de lock o variable de condición, se manipulan con:

-  `sem_wait()`: Decrementa el contador, si es  $< 0$  el hilo es enviado a dormir.
-  `sem_post()`: Incrementa el contador y despierta alguno de los hilos en espera.

#### **Semáforos como lock:**

- Se implementa un semáforo binario.
- Se inicializa en 1 de forma que se comporte como un lock..

#### **Semáforos como variable de condición:**

- Se inicializa en 0, de forma que este se comporte como un flag o indicador.

Valentina Cadena Zapata C.C 1000099120

Juan Manuel Vera Ospina C.C 1000416823