Concurrencia

> Hay 2 cosas complejas en ciencias de la computacion:
>
> 0. Poner nombres
> 1. La Concurrencia
> 2. Errarle por uno

Un thread nunca puede ser un proceso ligero, es mucho mas.
Concurrencia no es lo mismo que paralelismo, es la "ilusion" del paralelismo.
Paralelismo es hacer muchas cosas a la vez en simultaneo en el mismo instante. La concurrencia es la posibilidad de realizar varias cosas a la vez (que no quiere decir que pasen simultaneamente).

Nuestra programacion es puramente secuencial, nuestra cabeza no esta pensada para pensar tareas de forma que se ejecuten al mismo tiempo.

El concepto clave es escribir un programa como una *secuencia de ejecucion* o *thread* que interactuan y comparten datos. Es la unidad minima de ejecucion que se define conceptualmente como un *stream de ejecucion*.

Desde el punto de vista del programador se va a tener procesos y en cada procesos threads. En la realidad vamos a tener una cantidad fija de threads en ejecucion y el resto con otro estado.

Es una secuencia de ejecucion *atomica* que representa una *tarea planificable de ejecucion*.

Definiciones

- Secuencia de ejecucion atomica: Cada thread ejecuta una secuencia de instrucciones como lo hace un bloque de codigo en el modelo de programacion secuencial.

- Tarea planificable de ejecucion: El sistema operativo tiene injerencia sobre el mismo en cualquier momento.

Thread vs procesos

Un proceso es un programa en ejecucion con derechos restringidos.
Un thread no califica como programa.

Esta abstraccion se caracteriza por:

- Thread id
- un conjunto los valores de los registros
- stack propio
- una politica y prioridad de ejecucion
- un propio errno
- datos especificos del thread

1. One thread per process: un proceso con una unica secuencia de instrucciones ejecutandose de inicio a fin.

2. Many thread per process: un programa es visto como threads ejecutandose dentro de un proceso con derechos restringidos.

3. Many single-threaded processes: limitacion de algunos sistemas operativos que permitian varios procesos, pero cada uno con un unico thread, lo que implica que puede haber varios threads ejecutandose en kernel mode.

4. Many kernel threads: para aprovechar recursos, tambien el kernel puede ejecutar

Thread Scheduler

El thread cree que se esta ejecutando todo el tiempo, como los procesos.
El thread cree que toda instruccion se ejecuta inmediatamente despues de la anterior. En realidad el kernel puede cortar su ejecucion y reanudarla cuando quiera, lo que causa problemas de sincronizacion.

Estructura y ciclo de vida de un thread

El SO provee la ilusion de que cada thread se ejecuta en su propio procesador.

Hay dos estados, uno *per thread* y otro compartido entre thread.

Threads Control Block

Se guarda el puntero al stack del thread y una copia de sus registros en el procesador.

Variables compartidas del heap

- El codigo
- Variables globales
- Variables del heap

Estados de un thread

- Init: se esta creando
- Ready: Esta en una lista con todos los otros threas ready.
- Running
- Waiting: Es una lista, si un thead esta ahi es porque ya paso por la ready list. Esta esperando que otro termine.
- Finished: Una lista de threads terminados.