# UT1 - Introducción a la programación concurrente

**Procesos: Elementos de un proceso.**

* De manera simplificada, se puede definir un proceso como un programa en ejecución (instancias de un navegador web, un procesador de texto, máquina virtual de java...)
* Un proceso está compuesto por:
* Las instrucciones que se van a ejecutar.
* El estado del propio proceso.
* El estado de ejecución, principalmente recogido en registros del procesador.
* El estado de la memoria.
* Los procesos están entrando y saliendo constantemente del procesador. Se denomina contexto a toda la información que determina el estado de un proceso en un instante dado.
* Sacar un proceso del procesador para meter otro se conoce como cambio de contexto. Un cambio de contexto implica capturar el estado de la CPU y de sus registros, de la memoria y de la propia ejecución del proceso saliente para poder restaurar la información equivalente en el proceso entrante y poder continuar en el punto en el que este último abandonó el procesador en el cambio de contexto anterior.

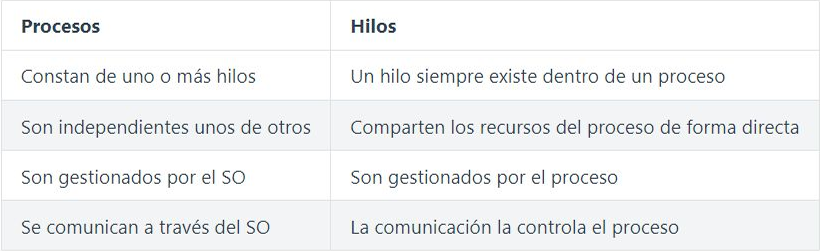
**Hilos: Concepto y características.**

* Un hilo o thread es cada una de las tareas que puede realizar de forma simultánea una aplicación. Por defecto, toda aplicación dispone de un único hilo de ejecución, al que se conoce como hilo principal. Si dicha aplicación no despliega ningún otro hilo, sólo será capaz de ejecutar una tarea al mismo tiempo en ese hilo principal.
* Así, para cada tarea adicional que se quiera ejecutar en esa aplicación, se deberá lanzar un nuevo hilo o thread. Para ello, todos los lenguajes de programación, como Java, disponen de una API para crear y trabajar con ellos.

**Hilos vs. procesos.**

* Un hilo es una secuencia de código en ejecución dentro del contexto de un proceso.
* Los hilos no pueden ejecutarse ellos solos.
* Requieren la supervisión de un proceso padre para correr.
* Dentro de cada proceso hay un hilo o varios hilos ejecutándose.
* La ventaja que proporcionan los hilos es la capacidad de tener más de un camino de ejecución en un mismo programa.

*Word puede tener un hilo en background chequeando automáticamente la gramática de lo que se escribe, mientras otro hilo puede estar guardando automáticamente los cambios del documento en el que se trabaja.*



**Programación concurrente y paralela**

Podemos decir que dos procesos son concurrentes cuando la primera instrucción de uno de los procesos se ejecuta después de la primera y antes de la última de otro proceso.

La planificación alternando los instantes de ejecución, multitarea, hace que los procesos se ejecuten de forma concurrente.

También la disponibilidad de varias unidades de proceso, multiproceso, permite la ejecución simultánea o paralela de procesos en el sistema.

**Programa secuencial**

Los programas secuenciales presentan una línea simple de control de flujo. Las operaciones de este tipo de programas están estrictamente ordenadas como una secuencia temporal lineal.

El comportamiento del programa es solo función de la naturaleza de las operaciones individuales que constituye el programa y del orden en que se ejecutan (determinado por el conjunto de entradas que recibe).

En los programas secuenciales, el tiempo que tarda cada operación en ejecutarse no tiene consecuencias sobre el resultado.

**Programa concurrente**

En los programas concurrentes existen múltiples líneas de control de flujo. Las sentencias que constituyen el programa no se ejecutan siguiendo un orden que corresponda a una secuencia temporal lineal.

En los programas concurrentes el concepto de secuencialidad entre sentencias continúa siendo muy importante. Sin embargo, en los programas concurrentes es de orden parcial, mientras que, tal y como hemos comentado anteriormente, en los programas secuenciales era de orden estricto.

En los programas concurrentes la secuencialización entre procesos concurrentes se llama sincronización.

**Comunicación y sincronización entre procesos**

Cuando varios procesos se ejecutan concurrentemente puede haber procesos que colaboren para un determinado fin mientras que puede haber otros que compitan por los recursos del sistema.

En ambos casos se hace necesaria la introducción de mecanismos de comunicación y sincronización entre procesos.

**Sistemas multitarea**

Tipos de sistemas multitarea:

* **Multitarea** **Nula**: SO que no dispone de multitarea, por ejemplo, MS-DOS.
* **Multitarea** **Cooperativa** o **no** **apropiativa**: Es aquella donde los procesos de usuario son los que ceden la CPU al SO a intervalos regulares. Un ejemplo de este tipo de multitarea son los Windows anteriores a 1995 como Windows 3.11.
* **Multitarea** **Preferente** o **apropiativa**: Es aquella en donde el SO se encarga de administrar los procesadores, repartiendo el tiempo de uso del mismo entre los procesos. En el caso de que solo se disponga de un procesador, se realizarán un número elevado de cambios de contexto, lo cual dará la sensación de que se ejecutan al mismo tiempo. Ejemplo: sistemas UNIX, Windows NT, etc.
* **Multitarea** **Real**: El SO ejecuta los procesos realmente al mismo tiempo, haciendo uso de múltiples procesadores. En el caso de que los procesos sean más que la cantidad de procesadores, se ejecutarán como en la multitarea preferente. Los SO dotados con esta capacidad son las variantes de Unix, Windows NT y Mac OS X.

**Sistemas distribuidos**

Posiblemente el ejemplo más famoso y conocido de sistema distribuido sea Internet. Internet aparece ante los usuarios como un enorme repositorio de documentos, es decir, como un único sistema capaz de proveer casi cualquier tipo de información o servicio que se necesite. No obstante, sabemos que está compuesta por millones de equipos ubicados en localizaciones diferentes e interconectados entre sí.

Nace de la necesidad de compartir recursos. Actualmente el máximo exponente de este tipo de sistemas es el Cloud Computing o servicios en la nube. Un sistema es distribuido cuando los componentes software están distribuidos en la red, se comunican y coordinan mediante el paso de mensajes.

Las características de este tipo de sistemas son:

* Concurrencia: ejecución de programas concurrentes.
* Inexistencia de un reloj global. Implica sincronizarse con el paso de mensajes.
* Fallos independientes: cada componente del sistema puede fallar sin que perjudique la ejecución de los demás

**Sistemas multitarea. ¿Para qué?**

Las principales razones por las que se utiliza una estructura concurrente son:

* Optimizar la utilización de los recursos: Podremos simultanear las operaciones de E/S en los procesos. La CPU estará menos tiempo ociosa.
* Proporcionar interactividad a los usuarios (y animación gráfica).
* Mejorar la disponibilidad: Servidor que no realice tareas de forma concurrente, no podrá atender peticiones de clientes simultáneamente.
* Conseguir un diseño conceptualmente más comprensible y mantenible: El diseño concurrente de un programa nos llevará a una mayor modularidad y claridad.

Además, los actuales avances tecnológicos hacen necesario tener en cuenta la concurrencia en el diseño de las aplicaciones para aprovechar su potencial. Los nuevos entornos hardware son:

* Microprocesadores con múltiples núcleos que comparten la memoria principal del sistema.
* Entornos multiprocesador con memoria compartida.
* Entornos distribuidos y servicios cloud.