UT1 - Introducción a la programación concurrente

Programación de servicios y procesos

Objetivos:

En este primer tema vamos a conocer los conceptos básicos relacionados con la programación concurrente, así como la mayoría de la terminología que vamos a trabajar y utilizar durante todo el curso.

En un mundo en el que cada vez los dispositivos electrónicos son cada vez más potentes, y veloces, el software debe ser capaz de aprovechar las características que le ofrecen tanto el hardware como los sistemas operativos.

Son muchas las tareas que requieren de un procesamiento rápido de cantidades ingentes de datos. Un par de ejemplos los tenemos en las aplicaciones Big Data e Inteligencia artificial. Estos dos campos son unos de los máximos exponentes en cuanto a programación concurrente.

Los objetivos que alcanzaremos tras esta unidad son:

- Diferenciar entre programa y proceso
- Comprender qué es la concurrencia
- Conocer el concepto, diferencias y relación existente entre las dos unidades básicas de ejecución: procesos e hilos.
- Tener nociones sobre programación concurrente
- Entender el funcionamiento concurrente del SO y del hardaware

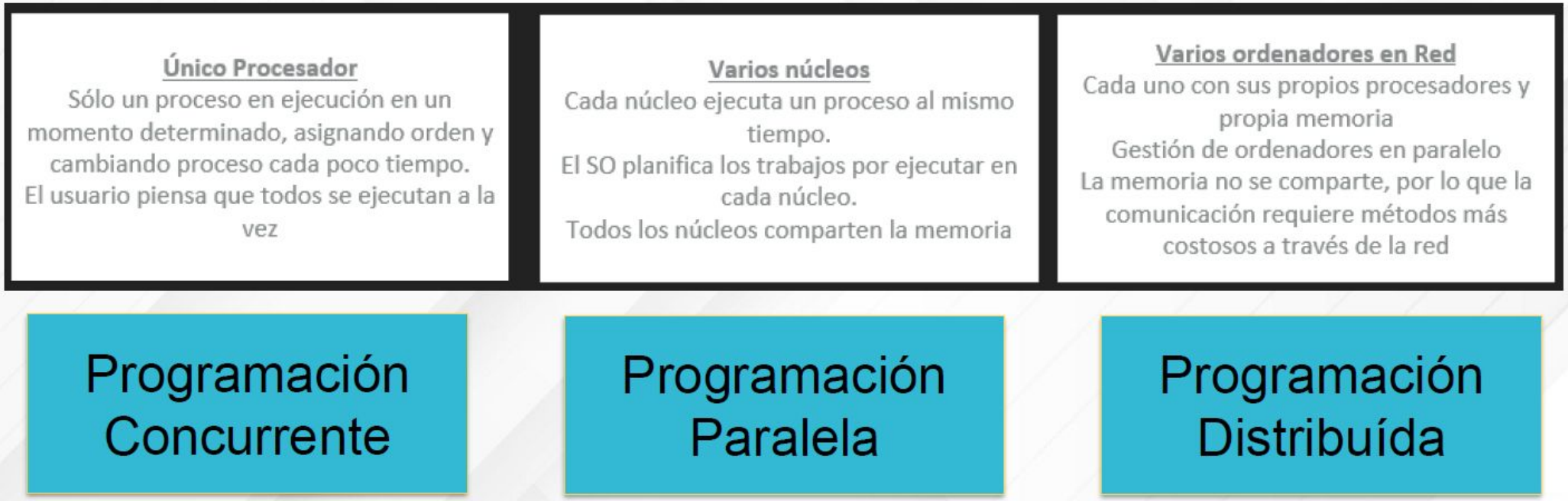
▶ ¿Qué es para ti concurrencia?

Contenidos:

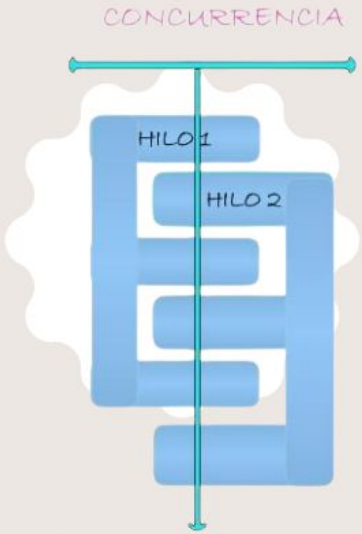
1) Introducción.
2) Programas. Ejecutables. Procesos. Servicios.
3) Procesos: Elementos de un proceso.
4) Estados de un proceso. Cambios de estado.
5) Hilos: Concepto y características.
6) Hilos vs. procesos.
7) **Sistemas multitarea: Programación concurrente. Programación paralela. Programación distribuida.**
8) **Planificación de procesos por el sistema operativo.**

Sistemas multitarea.

- Las tareas se pueden ejecutar de tres formas diferentes:

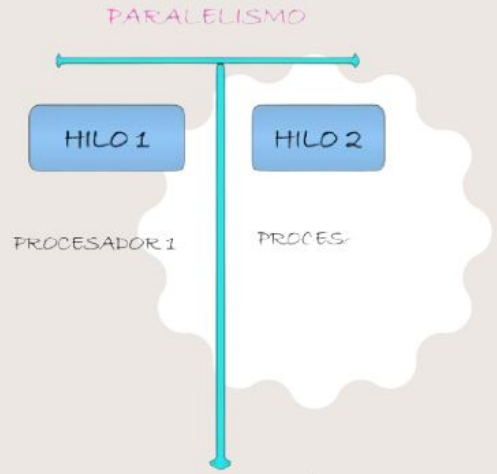


Sistemas multitarea.



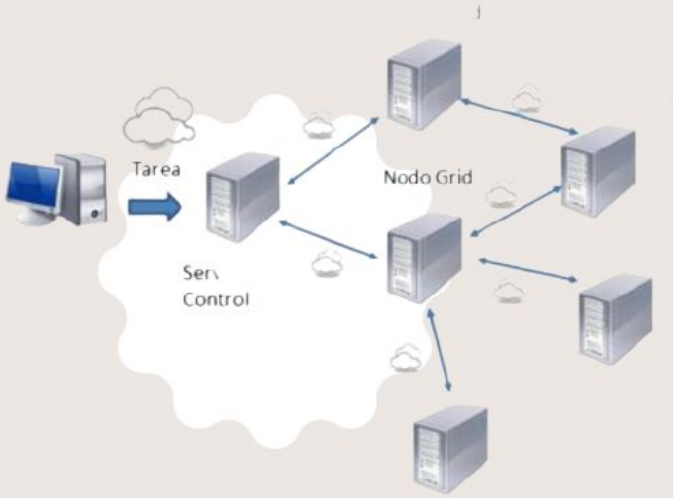
Programación concurrente

Es la programación de aplicaciones capaces de realizar varias tareas de forma simultánea utilizando hilos o threads. Hay que tener en cuenta que diferentes hilos pueden compartir información entre sí y eso complica mucho su programación y coordinación.



Programación paralela

Es la programación de aplicaciones que ejecutan tareas de forma paralela, de forma que no compiten por el procesador puesto que cada una de ellas se ejecuta en uno diferente. Normalmente buscan resultados comunes dividiendo el problema en varias tareas que se ejecutan al mismo tiempo.



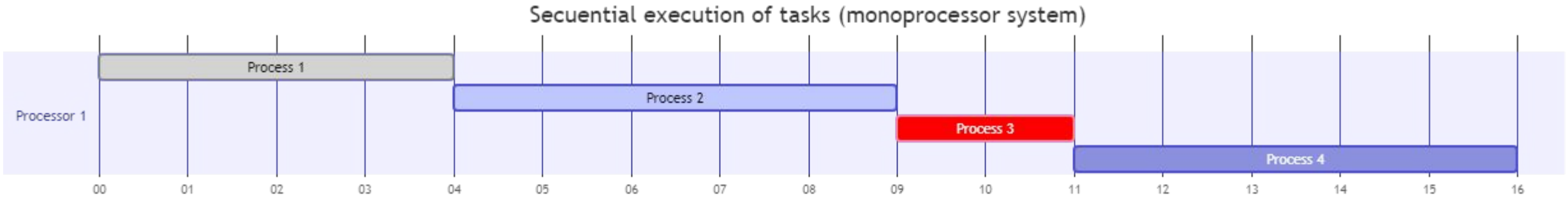
Programación distribuida

Es la programación de aplicaciones en las que las tareas a ejecutar se reparten entre varios equipos diferentes (conectados en red, a los que llamaremos nodos). Juntos, estos equipos, forman lo que se conoce como un Sistema Distribuido, que busca formar redes de equipos que trabajen con un fin común.

Sistemas multitarea.

Tipos de sistemas multitarea:

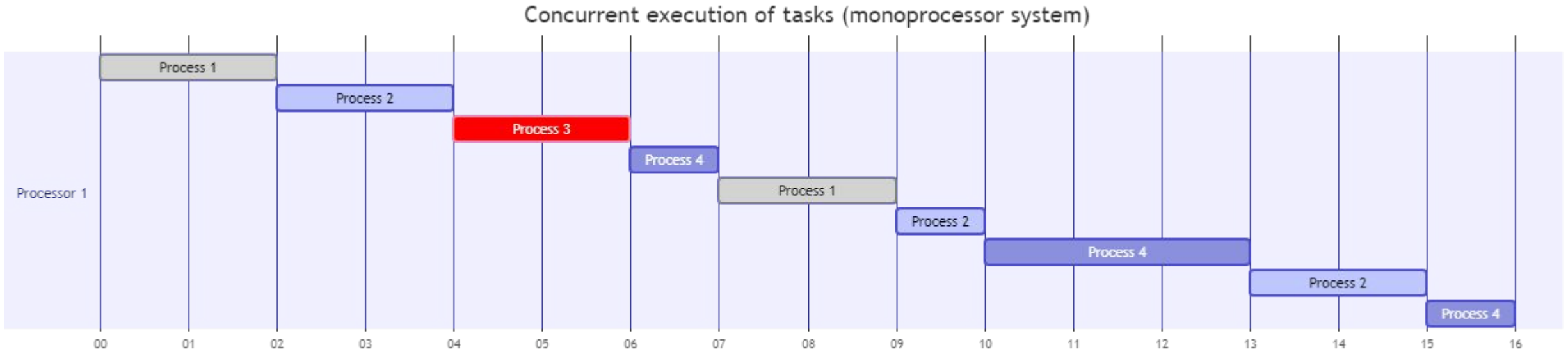
- **Multitarea Nula**: SO que no dispone de multitarea, por ejemplo, MS-DOS.



Sistemas multitarea.

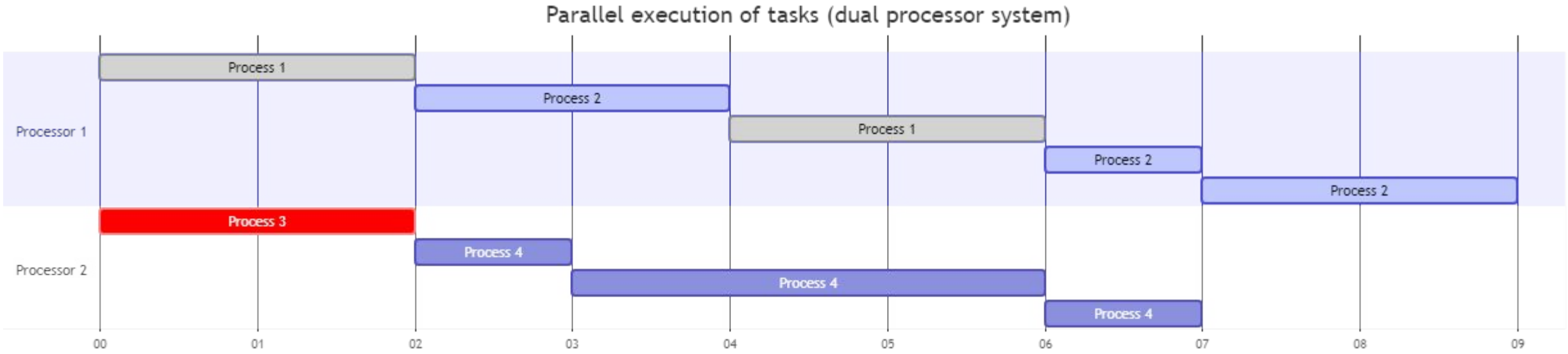
- **Multitarea Cooperativa o no apropiativa**: Es aquella donde los procesos de usuario son los que ceden la CPU al SO a intervalos regulares. Un ejemplo de este tipo de multitarea son los Windows anteriores a 1995 como Windows 3.11.

- **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 cuál dará la sensación de que se ejecutan al mismo tiempo. Ejemplo: sistemas UNIX, Windows NT, etc.



Sistemas multitarea.

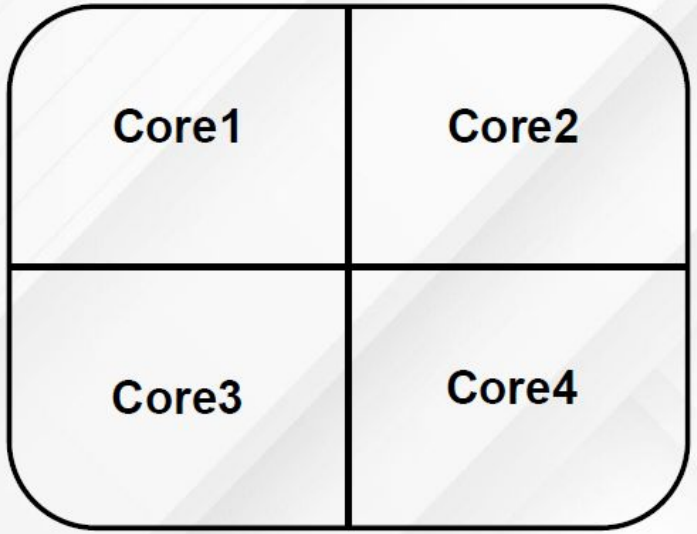
- **Multitarea Real**: El SO ejecuta los procesos realmente al mismo tiempo, haciendo uso de múltiples procesadores. En el caso de que los procesos sean más que la cantidad de procesadores, se ejecutarán como en la multitarea preferente. Los SO dotados con esta capacidad son las variantes de Unix, Windows NT y Mac OS X.



Sistemas multitarea.

Los cores (núcleos) son divisiones dentro de un procesador para poder repartir la carga que se le administra al procesador.

Cada uno de los cores funciona como una unidad independiente de proceso.



Ejemplo

Procesador Intel i7 6500U

Cantidad de núcleos: 4

Cantidad de subprocesos: 4

Sistemas distribuidos

un sistema distribuido es una colección de **computadores independientes** que aparecen ante los usuarios como un único sistema coherente.

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

Ejemplo de programación paralela y distribuida

Búsqueda de inteligencia extraterrestre - Proyecto SETI

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.

Procesos en el Sistema Operativo

El kernel o núcleo de un SO se encarga de la funcionalidad básica del sistema, el responsable de la gestión de los recursos del ordenador, se accede al núcleo a través de las llamadas al sistema, es la parte más pequeña del sistema en comparación con la interfaz. El resto del sistema operativo se le denomina como programas del sistema.

Todos los programas que se ejecutan en el ordenador se organizan como un conjunto de procesos. Es el sistema operativo el que decide parar la ejecución , por ejemplo, porque lleva mucho tiempo en la CPu, y decide cuál será el siguiente proceso que pasará a ejecutarse.

Cuando se suspende la ejecución de un proceso, luego deberá reiniciarse en el mismo estado en el que se encontraba antes de ser suspendido. Esto implica que debemos almacenar en algún sitio la información referente a ese proceso para poder luego restaurarla tal como estaba antes. Esta información se almacena en el **PCB (Bloque de control de procesos)**.

Gestión de procesos. Cambios de contexto

La CPU está constantemente ejecutando instrucciones independientemente del proceso, por lo tanto, ¿cómo ejecuta instrucciones de un proceso, otro,…?
Se guarda una "imagen" (**contexto**) del proceso en curso y se pasa a ejecutar otro.

Un **contexto** comprende:

- Estado del proceso.
- Estado del procesador.
- Información de gestión de memoria.

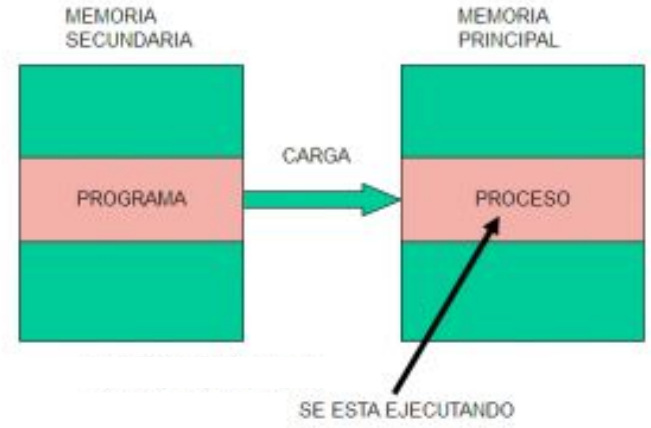
Cuando un proceso es *interrumpido* (planificación, liberación de memoria,..) por el sistema, se realiza esta salvaguarda de información.

Un cambio de contexto es tiempo perdido, ya que el procesador no hace trabajo útil durante ese tiempo y duración depende de la arquitectura del procesador.

Los cortes de ejecución son llevados a cabo dentro de la *Unidad de Control*, por lo tanto, se tienen que gestionar correctamente el contador de programa y el puntero a pila. Estos registros gestionan las direcciones de memoria de la siguiente dirección a ejecutar y el contexto del proceso.

Estados de un proceso. Cambios de estado.

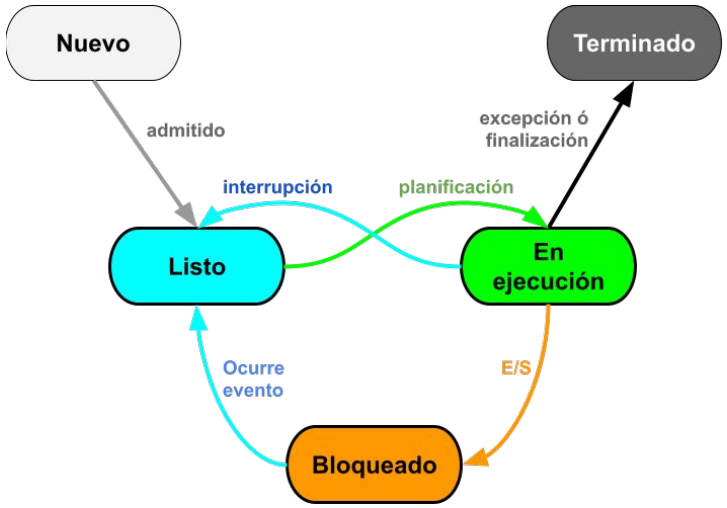
- Un procesador es capaz de realizar muchas tareas, aunque nosotros apreciaremos que solo está ejecutando la aplicación que nosotros estamos utilizando.
- Al procesador, lo único que le importa es ejecutar instrucciones y dar sus resultados independientemente del proceso que se esté ejecutando.
- ¿En qué memoria se ejecuta un programa? ¿En qué memoria se ejecuta un proceso? ¿Está siempre ejecutándose un proceso? En caso negativo, ¿qué hará?

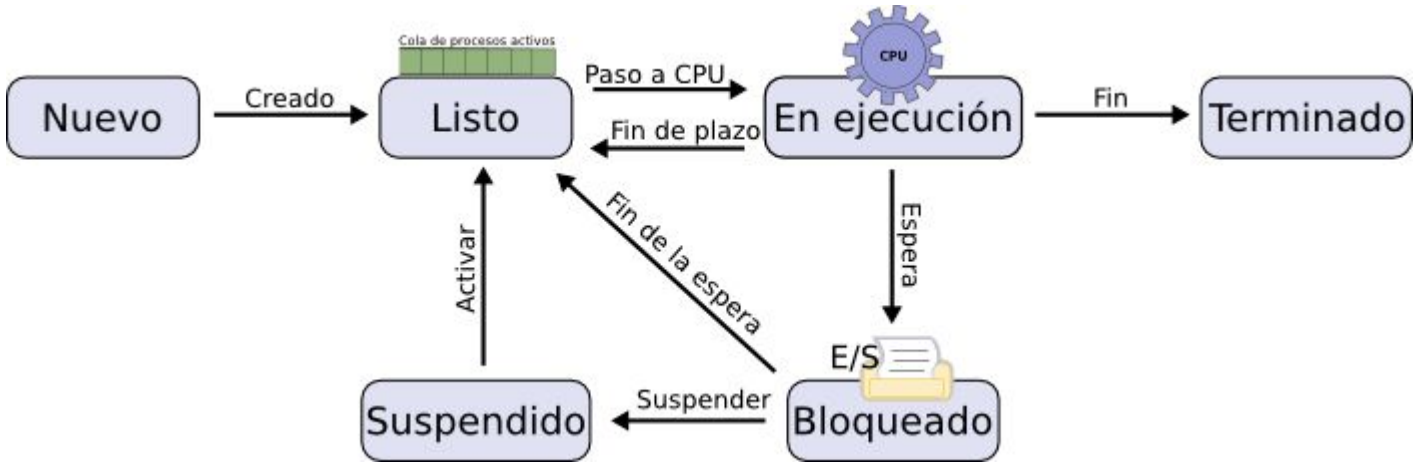


Estados de un proceso. Cambios de estado.

Podemos definir el conjunto de estados por los que pasa un proceso:

- **Nuevo :** Proceso nuevo, creado. Ejemplo: Inicialización del sistema.
- **Listo o ready:** Proceso que está esperando la CPU para ejecutar sus instrucciones.
- **En ejecución o run:** Proceso que actualmente se está ejecutando en la CPU.
- **Bloqueado o wait:** Proceso que está a la espera de la ejecución un evento externo, ejemplo, señal de E/S.
- **Suspendido**: Proceso que se ha llevado a la **memoria virtual o swap** para liberar memoria principal.
- **Terminado:** Proceso que ha finalizado y ya no necesitará más la CPU.





Planificación de procesos por el sistema operativo.

Uno de los objetivos de los sistemas operativos es la multiprogramación, es decir, admitir varios procesos en memoria para maximizar el uso del procesador. Esto funciona ya que los procesos se irán intercambiando el uso del procesador para su ejecución de forma concurrente. Para ello, el sistema operativo organiza los procesos en varias colas pasándolos de unas colas a otras

- Cola de procesos: contiene todos los procesos del sistema
- Cola de procesos preparados: todos los procesos listos esperando para ejecutarse.
- Varias colas de dispositivos: procesos que están esperando alguna operación de E/S.

Planificación de procesos por el sistema operativo.

El planificador de procesos es el elemento del sistema operativo que se encarga de repartir los recursos del sistema entre los procesos que los demandan. De hecho, es uno de sus componentes fundamentales, ya que determina la calidad del multiproceso del sistema y, como consecuencia, la eficiencia en el aprovechamiento de los recursos.

- Debe ser imparcial y eficiente.
- Debe minimizar el tiempo de respuesta al usuario, sobre todo en aquellos procesos o tareas más interactivas.
- Debe ejecutar el mayor número de procesos.
- Debe mantener un equilibrio en el uso de los recursos del sistema.

Existen diferentes algoritmos de planificación, cada uno con sus ventajas e inconvenientes.