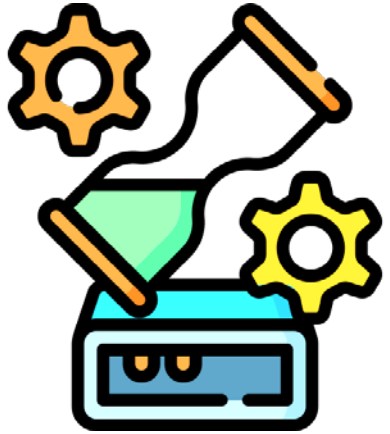
UT3.2: Gestión de los recursos de un SO: Los Procesos



FONDO SOCIAL EUROPEO

Javier Traseira

Conceptos clave

Algunos **conceptos clave** usados por los Sistemas Operativos son:

- **Usuario:** Toda aquella persona que trabaja en el sistema.
- **Sesión:** Periodo de tiempo durante el cual un usuario interactúa con el sistema.
- **Programa:** Código ejecutable. Se trata de un concepto estático
- **Proceso:** Programa en ejecución y en memoria. Concepto dinámico.
- **Fichero:** Unidad lógica de almacenamiento de datos.
- **Programas del sistema:** Ofrecen un entorno proporcionado por el SO para el desarrollo y ejecución de programas.
- **Interfaz de usuario:** Permite dar instrucciones al SO a través de diversas formas.

Procesos

Un **proceso** es un concepto manejado por el sistema operativo y que hace referencia a un conjunto de instrucciones de un programa en ejecución cargado en memoria.

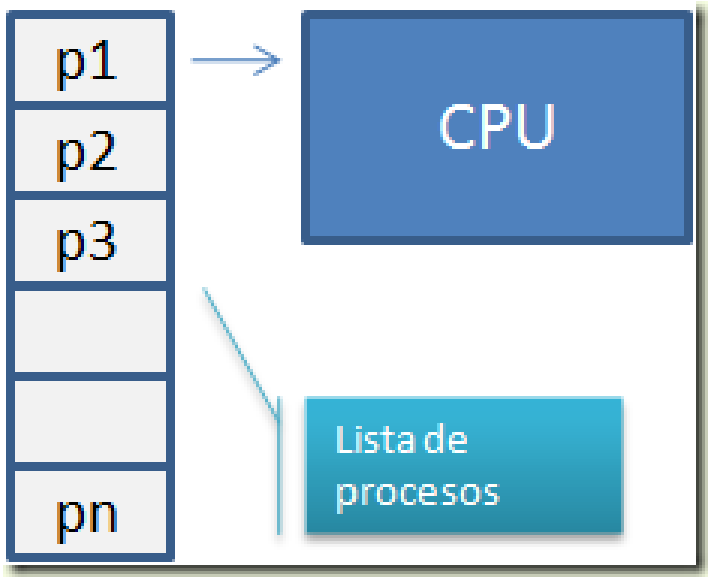
A los procesos, dependiendo del sistema operativo utilizado, se les denomina también ***flujos de control***, ***tareas***, ***hebras o hilos (threads)***, dependiendo del contexto.

Los sistemas operativos son los encargados de gestionar los recursos de hardware requeridos (principalmente el uso de la CPU o E/S), atendiendo a la diferentes prioridades cuando se ejecuta más de un proceso de forma concurrente (en SO multitarea).

Cuando un proceso se carga en memoria, el sistema operativo le asigna información en el **bloque de control de procesos (BCP)**, el cual estudiaremos más adelante durante este tema.

Procesos

Si trabajásemos con un **sistema operativo monotarea** (como *MS-DOS* en su día) la gestión de procesos sería muy sencilla: la CPU ejecutaría todas las instrucciones del proceso de un programa hasta finalizar y solo en ese momento continuaría con otro proceso en cola si lo hubiera.



El problema de este tipo de sistemas se hace evidente ya que mientras no finaliza la ejecución de un programa no puede pasarse a otro desaprovechando recursos y tiempo.

Procesos

En la actualidad la inmensa mayoría de sistemas operativos son **multitarea**.

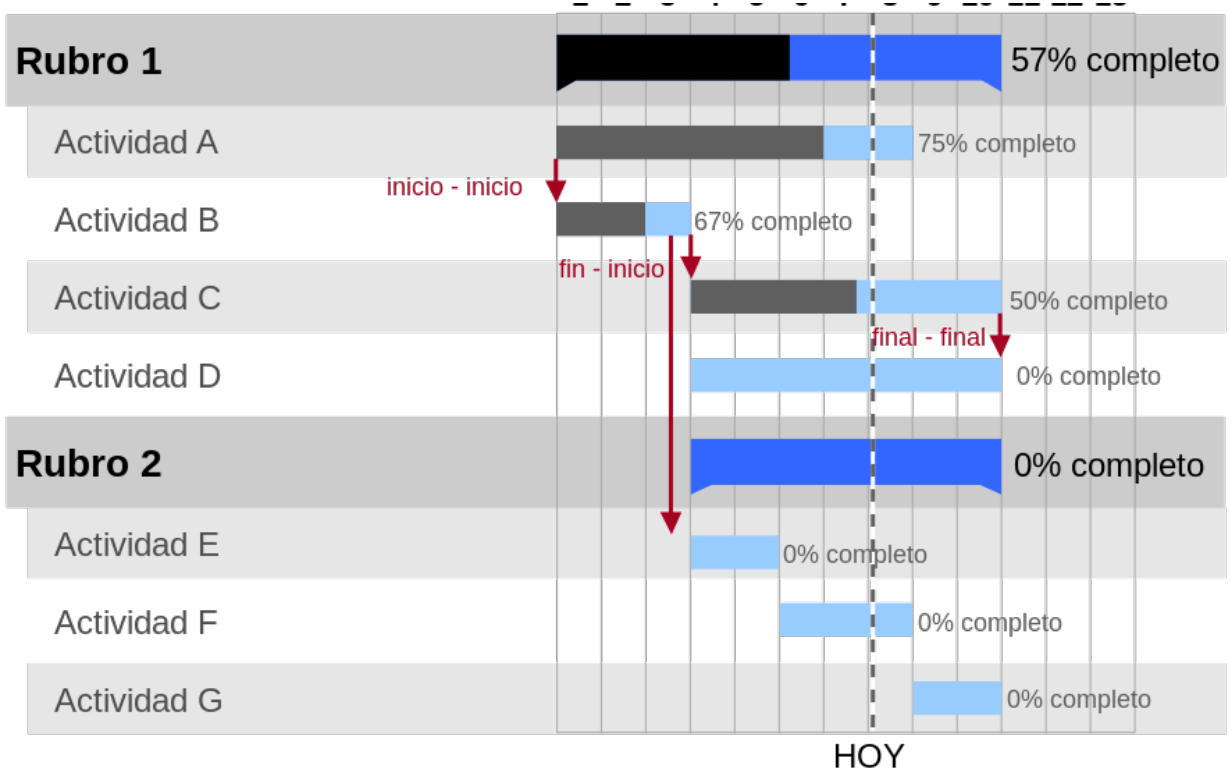
El sistema operativo esta hecho para no permitir a los procesos ejecutarse sino por una fracción de tiempo **muy pequeña**, de tal forma que en un solo segundo muchos programas han tenido tiempo de procesamiento, es decir muchos procesos de distintos programas se han ejecutado en lapsos diminutos de tiempo.

En un SO multitarea se aprovechan los tiempos de espera entre recursos y CPU de forma que esta se mantiene siempre trabajando. Esta técnica se conoce como **multiprogramación** y tiene como finalidad conseguir un mejor aprovechamiento de la CPU

Procesos

Para la resolución de ejercicios prácticos/teoría usaremos un tipo de diagramas ampliamente utilizado en informática denominados **diagramas de Gantt.**

Un **diagrama de Gantt** es una representación gráfica para representar el tiempo de dedicación previsto para diferentes tareas o procesos a lo largo de un tiempo total determinado.



Procesos

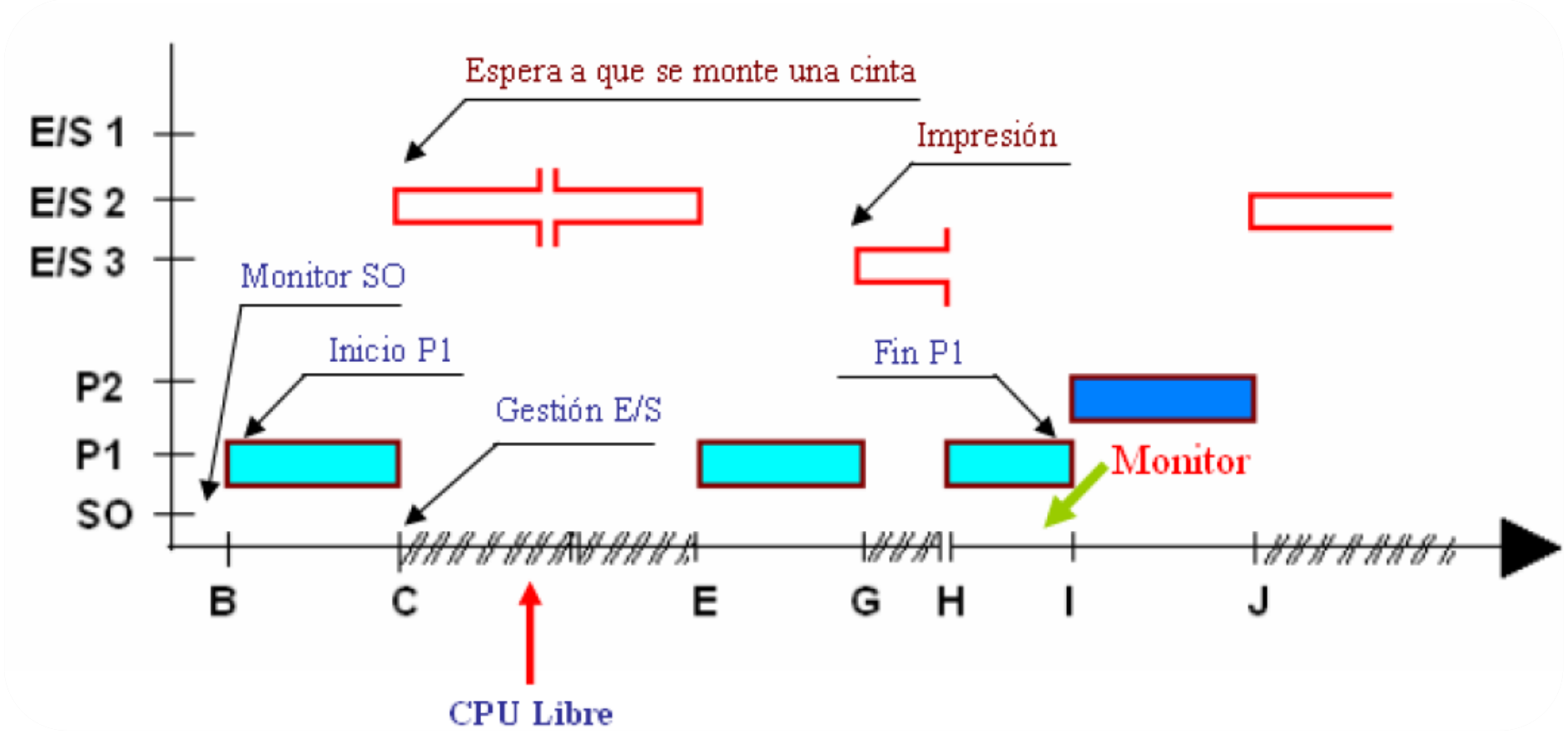


Diagrama de Gantt de la ejecución de dos procesos **P1** y **P2** en un SO monotarea.

El tiempo ***t*** en este tipo de diagramas se mide en **ciclos de procesador**

Procesos

Hilo (thread)

Se denomina **hebra, hilo** o thread a la unidad básica en la que se puede subdividir un proceso de forma concurrente. Un proceso tendrá siempre un hilo principal, pero puede tener más hilos que correrán en paralelo.

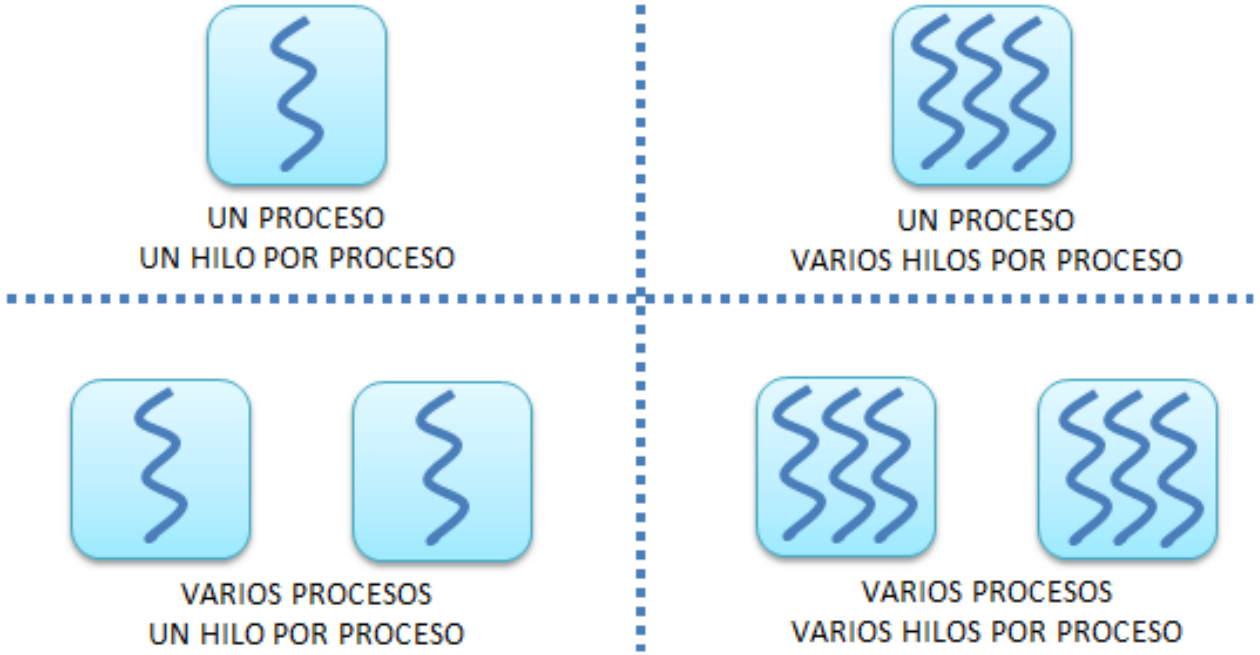
Los hilos dentro de una misma aplicación comparten: código y datos recursos del SO. (ficheros, E/S, etc.).

Un proceso clásico es aquel que solo posee un hilo.
Si por ejemplo ejecutamos un procesador de textos como Word, con un solo documento abierto, el programa Word convertido en proceso estará ejecutándose en un único espacio de memoria (con acceso a archivos, galerías de imágenes, corrector ortográfico..). Este proceso, de momento, tendrá un hilo. Si en esta situación, sin cerrar Word abrimos un nuevo documento, Word no se volverá a cargar como proceso. Simplemente el programa, convertido en proceso, tendrá a su disposición **dos hilos** o hebras diferentes, de tal forma que el proceso sigue siendo el mismo (el original). Word se está ejecutando una sola vez y el resto de documentos de texto que abramos serán hilos del principal.

Procesos

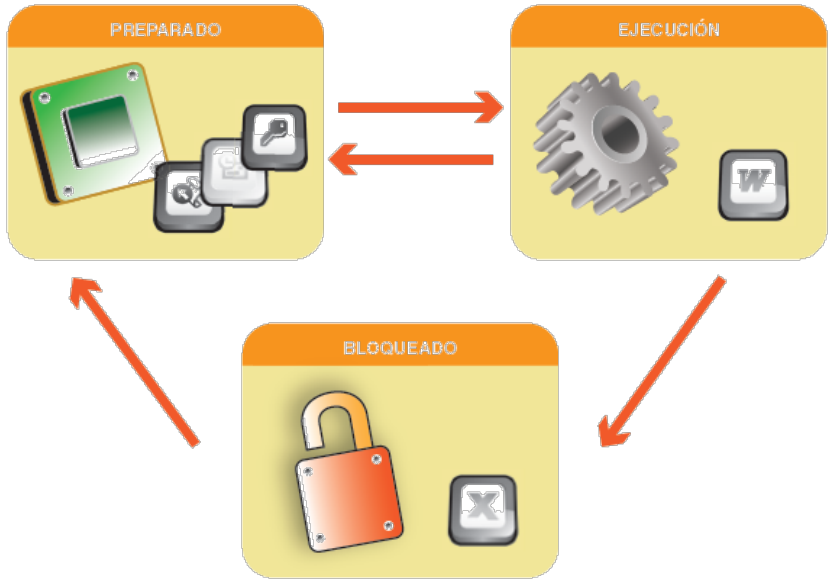
Hilo (thread)

La creación de un nuevo hilo es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.



Estados y transiciones de los procesos

Existen **tres estados** para los procesos (o hilos correspondientes):



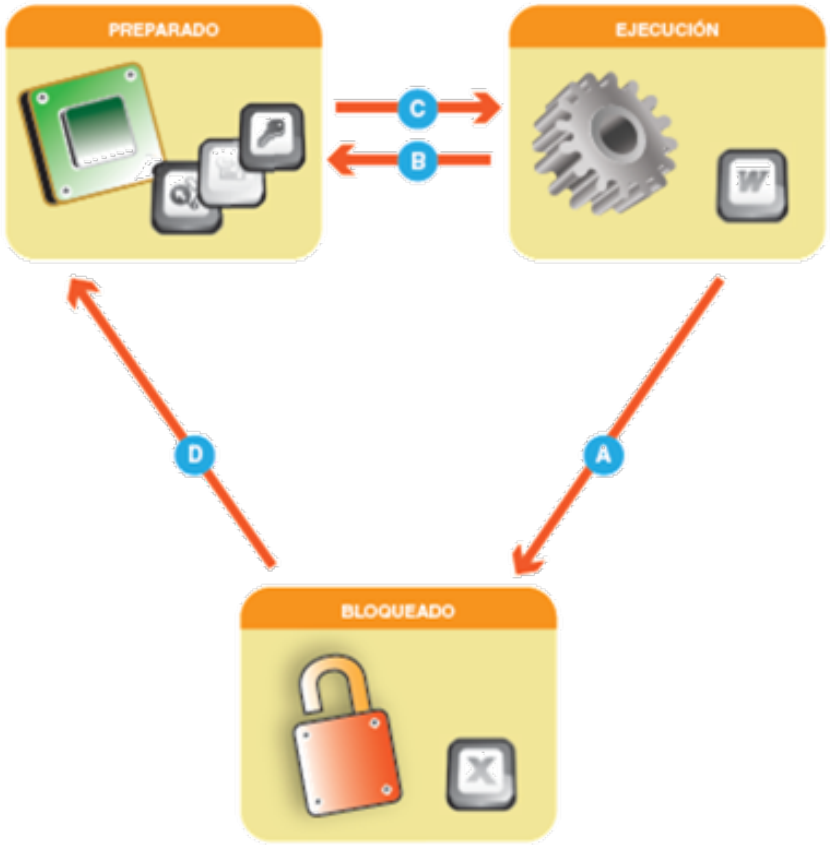
- **En ejecución:** El procesador está ejecutando instrucciones del proceso cargado en ese momento (tiene su atención y prioridad)
- **Preparado, en espera o activo:** El proceso está preparado para ser ejecutado y esperando su turno para ser atendido por la CPU.
- **Bloqueado:** El proceso ha entrado en un estado de bloqueo que puede darse por causas múltiples (acceso a un mismo fichero, errores..)

En algunas biografías pueden usarse también los estados **nuevo** y **terminado**

Estados y transiciones de los procesos

Una vez que un programa se ha lanzado y se ha convertido en proceso, puede atravesar varias fases o **estados** hasta que termina.

Los cambios de estado en los que se puede encontrar un proceso es lo que se denomina **transiciones**:



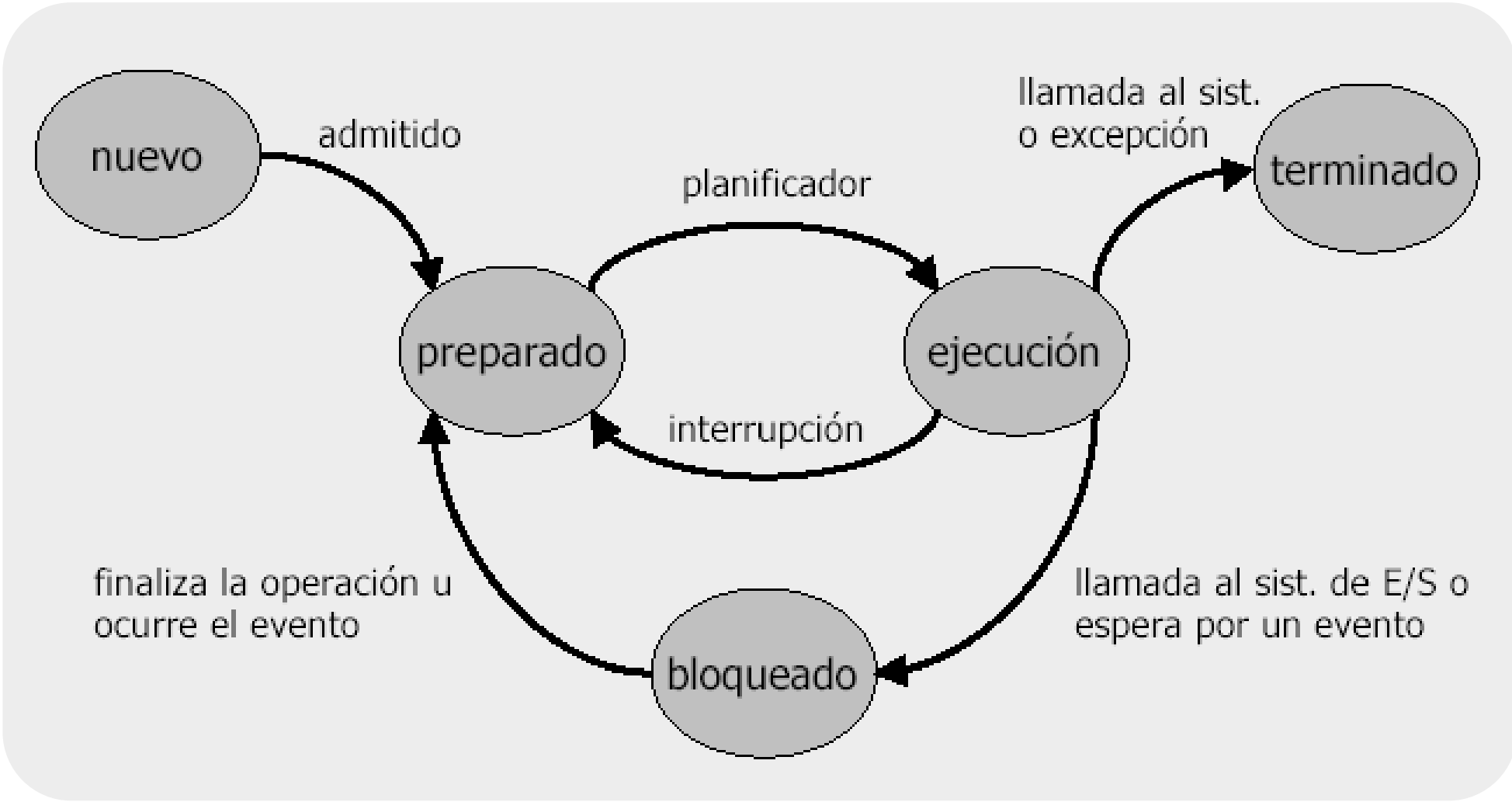
Transición A. Ocurre porque el proceso que está en ejecución necesita algún elemento, señal, dato, para poder continuar ejecutándose.

Transición B. Ocurre cuando un proceso ha utilizado el tiempo asignado por la CPU y deja paso al siguiente proceso.

Transición C. Ocurre cuando el proceso que está preparado pasa a estado de ejecución en la CPU.

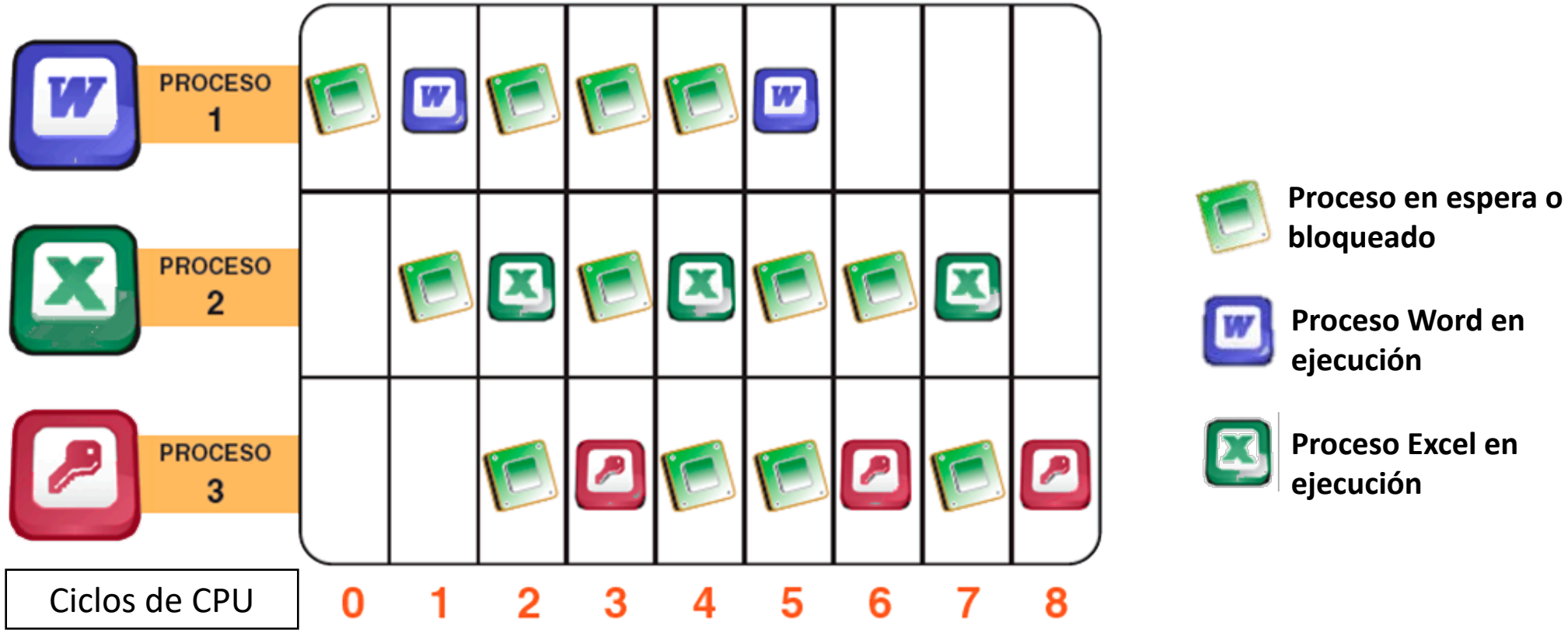
Transición D. Ocurre cuando el proceso pasa a preparado, es decir, al recite la orden o señal que estaba esperando en estado de bloqueado.

Estados y transiciones de los procesos



Estados y transiciones de los procesos

✓ En el siguiente diagrama observamos tres procesos (*o hilos*) pasando de estado de ejecución a quedar en espera o bloqueados:



Estados y transiciones de los procesos

Del que un proceso cambie de estado en un momento u otro se encarga el **planificador de procesos del sistema operativo**.

El **planificador** de un sistema operativo se encarga de asignar **prioridades** a los diferentes procesos para llevar a cabo su ejecución en el menor tiempo y de la forma más óptima posible.

Mediante técnicas que veremos a continuación, se consigue indicar a la CPU del ordenador que procesos deben ejecutarse en qué momento concreto y los diferentes estados que deben ir adoptando. Ello se lleva cabo mediante **algoritmos de planificación**.

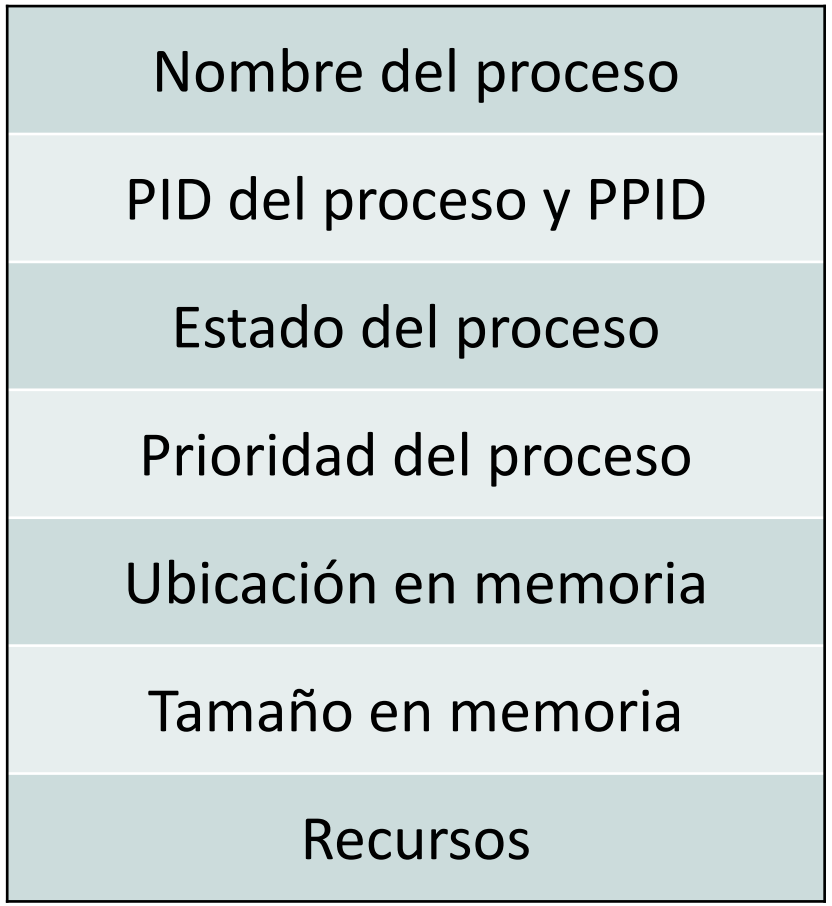
Bloque de control de procesos

La información de un proceso que el sistema operativo necesita para controlarlo se guarda en un **bloque de control de procesos o BCP**.

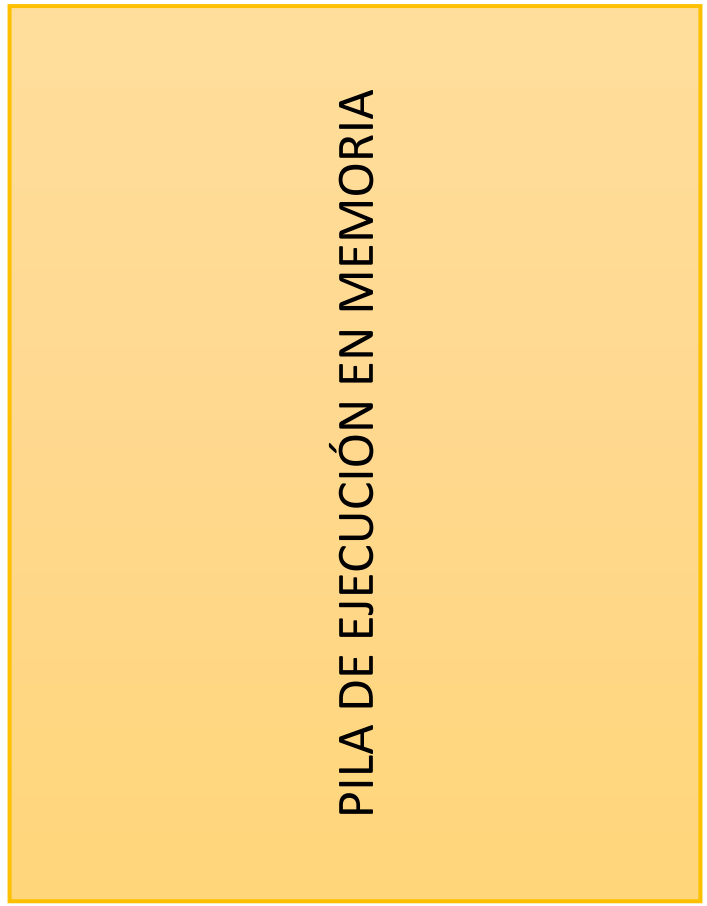
En el **BCP** cada proceso almacena información como:

- **Nombre del proceso**
- **Identificador del nombre e identificador del proceso**. A cada proceso se le asigna un identificador denominado ***PID***. Si tiene un proceso padre se identificará a su vez con su ***PPID***.
- **Estado actual del proceso**: Ejecución, preparado o bloqueado.
- **Prioridad del proceso**. Se la asigna el planificador o el usuario de forma manual.
- **Ubicación y tamaño usado en memoria**. Dirección de memoria en la que está cargado el proceso y espacio utilizado.
- **Recursos utilizados**. Otros recursos hardware y software para poder ejecutarse.

Bloque de control de procesos



BCP básico de un proceso



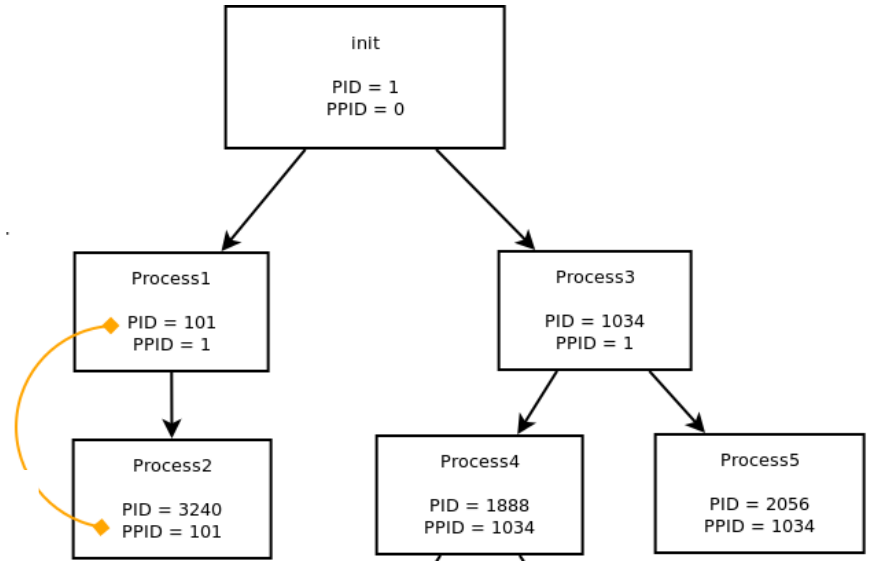
Pila de estado en memoria

Bloque de control de procesos

Los procesos se marcan en su creación con un número único llamado identificador de proceso (**PID**). Salvo el proceso raíz, todos los procesos llevan dos números:

- El *PID* que lo identifica a él.
- El PPID que identifica a su padre.

Después de que un proceso genera un hijo, ambos continúan ejecutándose desde el punto en el que se hizo su creación.



Bloque de control de procesos

Nombre	PID	Estado	Prioridad	Ubicación	Recursos
Taskmgr.exe	00014	Ejecución	Auto	0x773D	CPU
Notepad.exe	00015	Preparado	Auto	0xF223	Memoria
Dropbox.exe	00016	Preparado	Baja	0x7723	Memoria
Searchui.exe	00009	Bloqueado	Baja	0x006E	Memoria, E/S

- ✓ Abre el ***administrador de tareas*** en Windows y busca el detalle de al menos 5 procesos en el sistema. Apunta en una tabla su nombre, PID, estado y uso de memoria.

En Linux también pueden verse utilizado el comando *ps –aux* o el gnome system monitor dependiendo de la distribución.

Algoritmos de planificación

¿Qué es un algoritmo?

Un **algoritmo** es una serie ordenada de instrucciones o pasos o que llevan a la solución de un determinado problema.

Los hay tan sencillos y cotidianos como seguir la receta del médico, abrir una puerta, lavarse las manos, etc; hasta los que conducen a la solución de problemas muy complejos.

1. Tomar el cepillo de dientes
2. Aplicar crema dental al cepillo
3. Abrir el grifo
4. Remojar el cepillo con la crema dental
5. Cerrar el grifo
6. Frotar los dientes con el cepillo
7. Abrir el grifo
8. Enjuagarse la boca
9. Enjuagar el cepillo
10. Cerrar el grifo

Algoritmos de planificación

Gracias a los **algoritmos de planificación** usados en SO multiproceso, la CPU se encarga de asignar tiempos de ejecución a cada proceso según el tipo de algoritmo y la **prioridad** de cada proceso.

El objetivo de un algoritmo de planificación es decidir qué proceso se ejecuta en cada momento en la CPU de un SO multitarea.

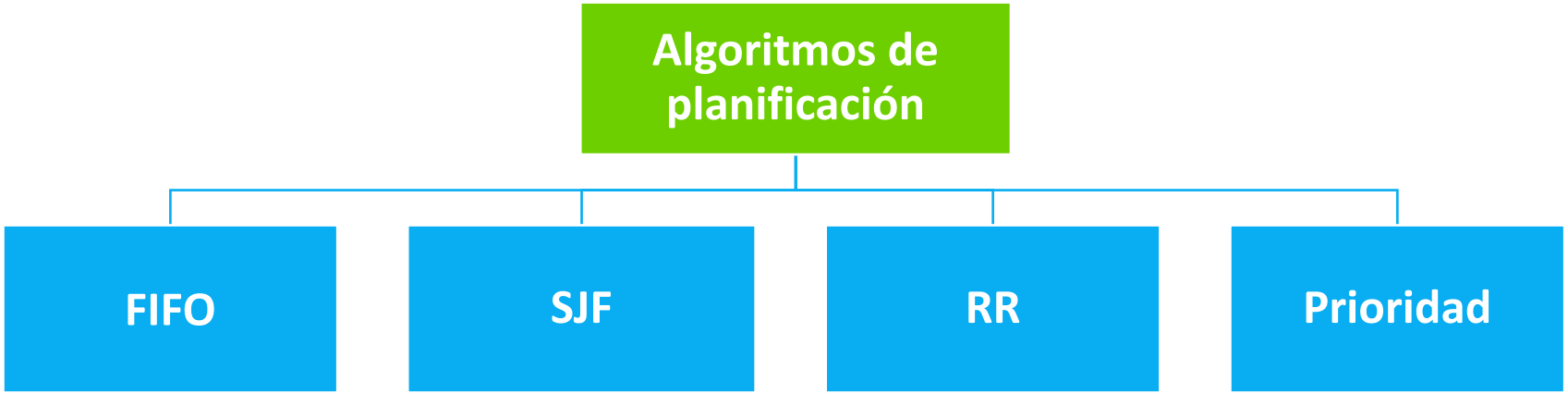
Dichos algoritmos pueden ser de dos tipos:

- **Apropiativos**: un proceso puede ser interrumpido por otro para dejarle acabar
- **No apropiativos**: una vez un proceso entra en la CPU no se libera hasta terminar.

Algoritmos de planificación

Los algoritmos de planificación usados en SO actuales que veremos son:

- **Algoritmo FIFO** (*First Input First Output*)
- **Algoritmo SJF** (*Shortest Job First*)
- **Algoritmo de rueda o RR** (*Round Robin*)
- **Algoritmo basado en prioridad**

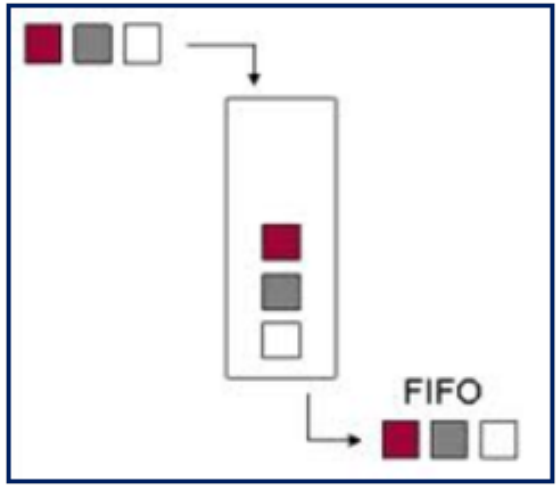


Algoritmos de planificación

Algoritmo FIFO

Para este algoritmo denominado **FIFO** (First Input, First Output), *el primero que entra es el que sale*. El procesador ejecuta cada proceso hasta que finaliza, por tanto, los procesos llegará a una cola de procesos a esperar en orden a que les llegue su turno.

Se trata de una política muy simple y sencilla de llevar a la práctica, pero de rendimiento pobre.



La cantidad de tiempo de espera de cada proceso depende del número de procesos que haya antes en cola de espera. Sus características son:

- ***No apropiativo*** (**La CPU no se libera hasta haber terminado**)
- Es justo, aunque los procesos largos hacen esperar mucho a los cortos.
- El tiempo medio de servicio es muy variable en función del número de procesos y su duración.

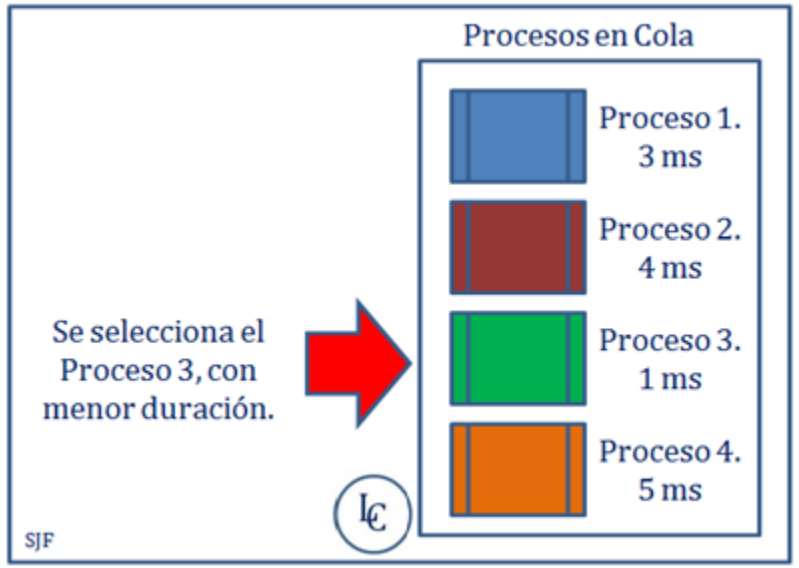
Algoritmos de planificación

Algoritmo SJF

El algoritmo **SJF** (Shortest Job First) que viene de 'el trabajo más corto primero'. Se trata de un algoritmo que supone que los tiempos de ejecución ya se conocen de antemano, algo que no siempre es posible saber. Cuando hay varios trabajos de igual importancia esperando a ser iniciados en la cola de entrada, el planificador seleccionará el trabajo más corto primero.

Sus características son:

- ***No apropiativo***.
- Muy complicado de implementar, necesario predecir los tiempo de ejecución de los procesos con antelación.
- Se considera relativamente óptimo.



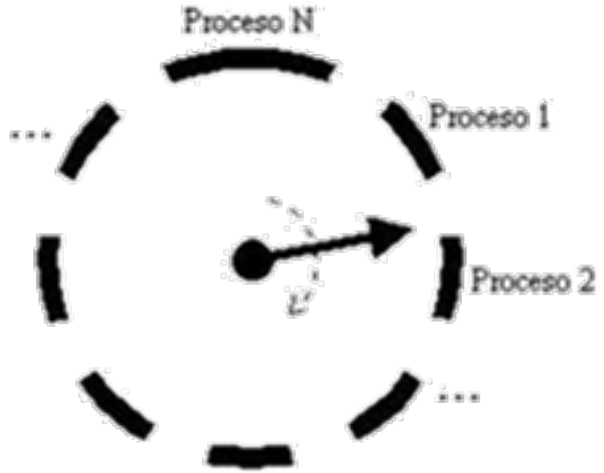
Algoritmos de planificación

Algoritmo RR (rueda)

El algoritmo **RR** (Round Robin) o de la rueda es uno de los algoritmos más sencillos y utilizados en sistemas Windows y Linux. En su utilización no se establecen prioridades (siendo apropiativo). Cada proceso tiene asignado un tiempo de ejecución denominado **quantum *(Q)***. Si se cumple ese tiempo y la tarea no ha concluido, se da paso al siguiente proceso y el proceso no finalizado pasa al final de la lista de procesos en espera.

Sus características son:

- ***Apropiativo***
- Es un algoritmo **justo** (evita la monopolización de la CPU)
- Su rendimiento depende del valor de ***Q***
- Usa FIFO para la gestión de la cola de procesos.

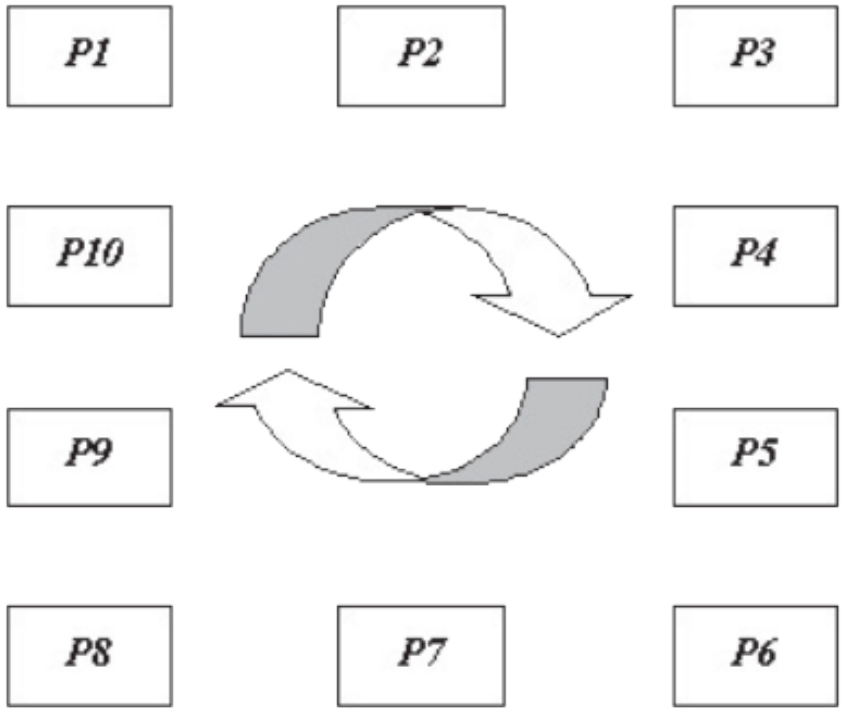


El *quantum* ***Q*** suele definirse entre unos *20ms* o *50ms*

Algoritmos de planificación

Algoritmo RR (rueda)

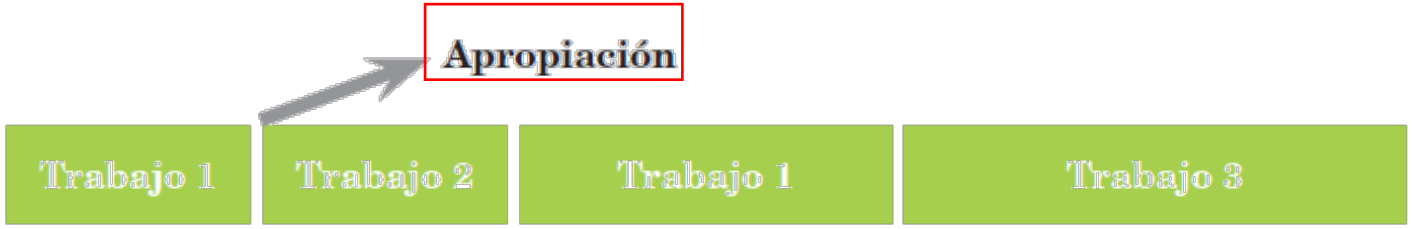
Tal y como se muestra en la figura, si se está ejecutando el *proceso 1* y se agota su cantidad de **quantum *Q***, se desalojaría la CPU y pasaría a ejecutarse el *proceso 2*. Cuando termine el quantum del *proceso 10* se pasará de nuevo al *proceso 1*.
Se usa **FIFO** a la hora de pasar de un proceso a otro en la cola.



Algoritmos de planificación

Algoritmo basado en prioridad

Otro tipos de algoritmos usados en SO modernos son los basados en **prioridades**. En ellos se asocia una prioridad a cada proceso y la CPU se asigna al trabajo con prioridad más alta en cada momento.
Normalmente, si se está ejecutando un proceso de prioridad media y entra un proceso de prioridad mayor, se requisa la CPU al primer proceso y se le entrega al proceso de mayor prioridad.



En el ejemplo anterior se observa que la CPU está ejecutando el trabajo 1 cuando se recibe la petición del trabajo 2 que tiene alta prioridad; entonces, se desaloja la CPU y se atiende la petición del trabajo 2 y, cuando termine, se recuperará el trabajo 1.

Algoritmos de planificación

Algunos conceptos importantes que usaremos a la hora de completar las tablas de los problemas de los diferentes algoritmos de planificación:

- **Ciclo de llegada:** Momento en el que llega un proceso al planificador de procesos del SO.
- **Ciclos de ejecución**: Ciclos de CPU que consume un proceso mientras se encuentra en estado de ejecución
- **Tiempo de espera**: Tiempo que un proceso está esperando en la cola de procesos preparados o listos.
- **Tiempo de retorno**: Tiempo que transcurre desde que un proceso llega, hasta que sale (tiempo que tarda en ejecutarse)

Algoritmos de planificación

1. Representa en forma de diagrama de Gantt del **algoritmo FIFO** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

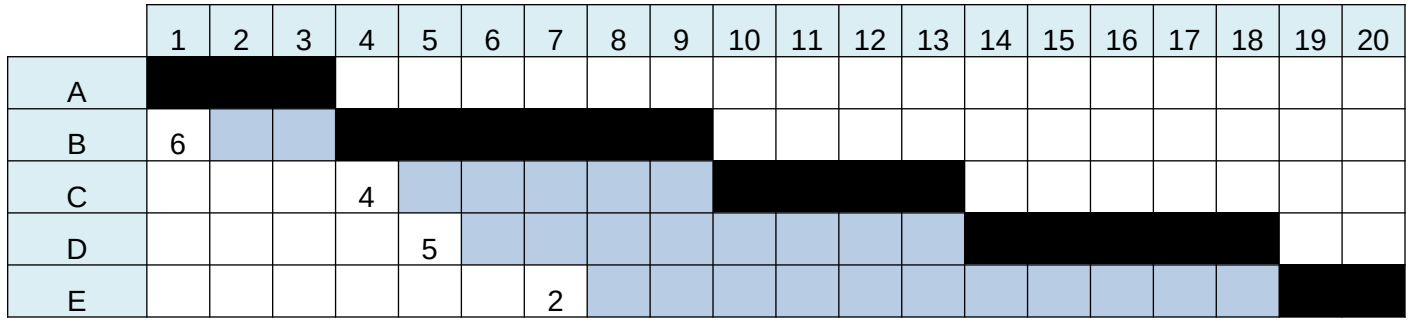
	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3				
B	2	6				
C	5	4				
D	6	5				
E	8	2				
				Tiempos promedio =		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B	6																			
C				4																
D					5															
E							2													

Algoritmos de planificación

1. Representa en forma de diagrama de Gantt del **algoritmo FIFO** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3	1	3	0	3
B	2	6	4	9	2	8
C	5	4	10	13	5	9
D	6	5	14	18	8	13
E	8	2	19	20	11	13
			Tiempos promedio =		**5,2**	**9,2**



Algoritmos de planificación

2. Representa en forma de diagrama de Gantt del **algoritmo SJF** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	8	2				
				Tiempos promedio =		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	■	■	■																	
B	6														■	■	■	■	■	■
C			4	■	■	■	■													
D					5					■	■	■	■	■						
E							2	■	■											

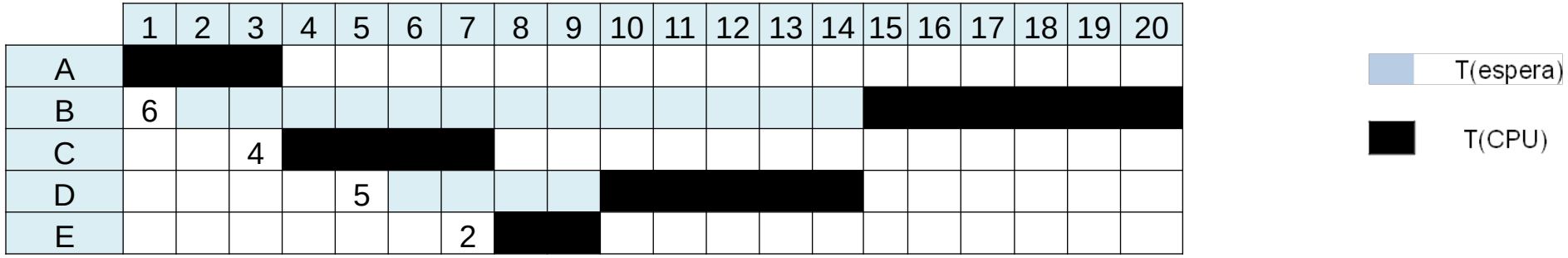
T(espera)

T(CPU)

Algoritmos de planificación

2. Representa en forma de diagrama de Gantt del **algoritmo SJF** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3	1	3	0	3
B	2	6	15	20	13	19
C	4	4	4	7	0	4
D	6	5	10	14	4	9
E	8	2	8	9	0	2
			Tiempos promedio =		**3,4**	**7,4**

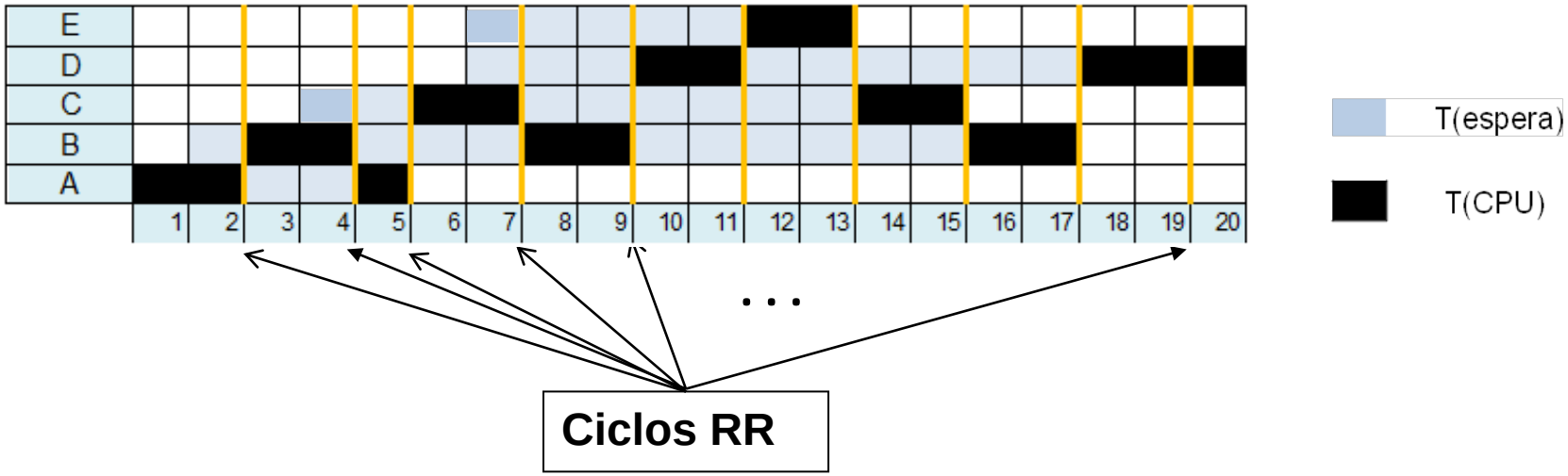


Algoritmos de planificación

3. Representa en forma de diagrama de Gantt del **algoritmo RR con q=2** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3				
B	2	6				
C	4	4				
D	6	5				
E	7	2				
				Tiempos promedio =		

Quantum = 2

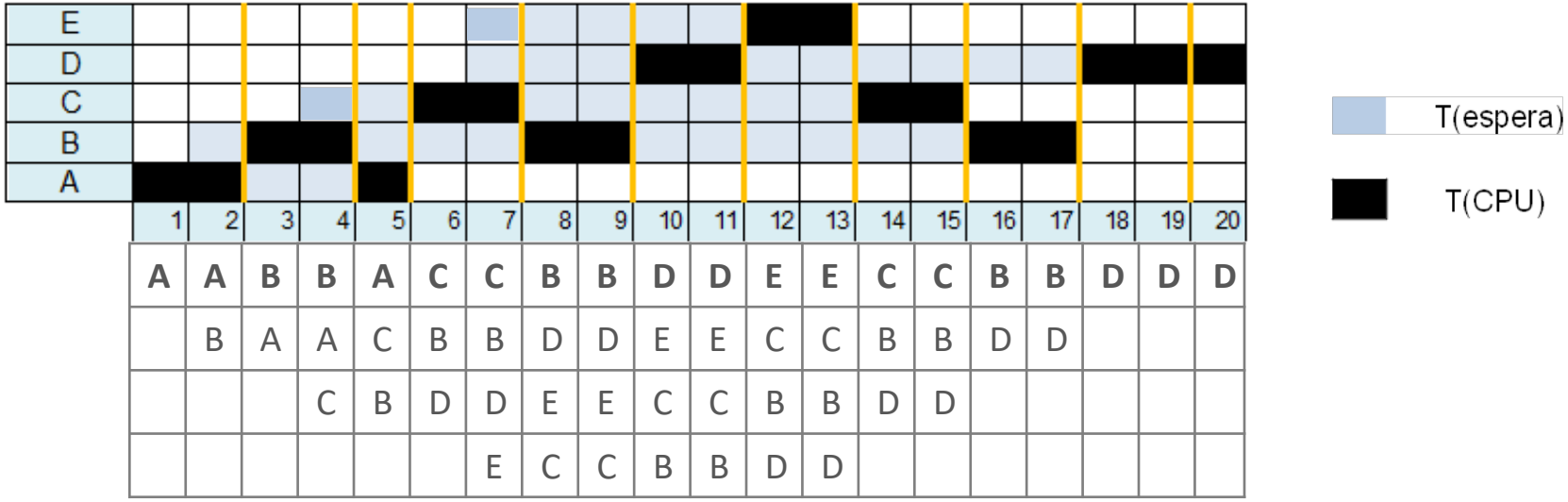


Algoritmos de planificación

3. Representa en forma de diagrama de Gantt del **algoritmo RR con q=2** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclos de ejecución	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3	1-5	5	2	5
B	2	6	3-8-16	17	10	16
C	4	4	6-14	15	8	12
D	6	5	10-18	20	10	15
E	7	2	12	13	5	7
			Tiempos promedio =		**7**	**10,8**

Quantum = 2



Algoritmos de planificación

4. Representa en forma de diagrama de Gantt del **algoritmo de Prioridad** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclo de ejecución	Prioridad	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3	**4**				
B	2	8	**2**				
C	4	1	**1**				
D	6	2	**3**				
E	9	6	**1**				
					Tiempos promedio =		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B	8																			
C			1																	
D					2															
E								6												

T(espera)

T(CPU)

Algoritmos de planificación

4. Representa en forma de diagrama de Gantt del **algoritmo de Prioridad** para la siguiente lista de 5 procesos (*A,B,C,D,E*):

	Ciclo de Llegada	Ciclo de ejecución	Prioridad	Ciclo inicial	Ciclo final	Tiempo de espera	Tiempo retorno
A	0	3	**4**	1 - 19	20	17	20
B	2	8	**2**	2 - 5 -15	16	7	15
C	4	1	**1**	4	4	0	1
D	6	2	**3**	17	18	11	13
E	9	6	**1**	9	14	0	6
				Tiempos promedio =		**7**	**11**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	■																		■	■
B	8	■	■		■	■	■	■							■	■				
C			1	■																
D					2												■	■		
E								6	■	■	■	■	■	■						

T(espera)

T(CPU)

Interrupciones y excepciones

Una **interrupción** es una señal que obliga al SO a tomar el control del procesador para estudiarla y tratarla.

Una interrupción es un mecanismo que permite ejecutar un bloque de instrucciones interrumpiendo la ejecución normal de un programa, y luego intentar restablecer la ejecución del mismo sin afectarlo directamente. De este modo un programa puede ser interrumpido temporalmente para atender alguna necesidad urgente del computador y luego continuar su ejecución como si nada hubiera pasado. Algunos tipos de interrupciones **no son recuperables**.

Existen varios tipos de interrupciones:

- Interrupciones de hardware
- Interrupciones de software.
- Excepciones.

Interrupciones y excepciones

Interrupción de hardware o software

Existen varios tipos de interrupciones, dependiendo de dónde se produzcan dichos eventos para ser atendidos:

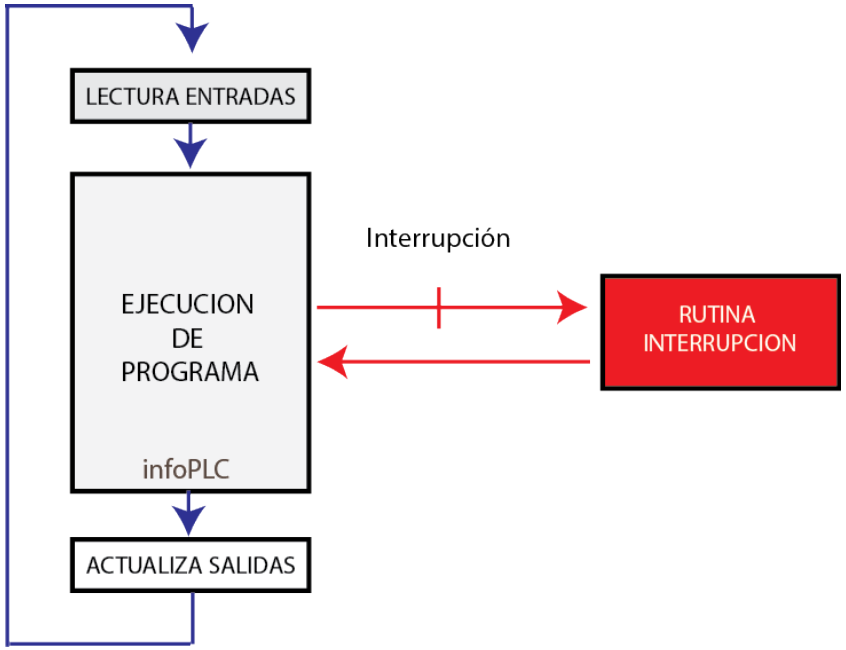
- **De hardware**, un dispositivo (hardware) requiere la atención de la CPU para ejecutar su driver. Suelen aparecer de forma no programada (inesperadas)
  - **De I/O**, provocadas por los dispositivos de I/O.
  - **Externas**, provocadas por elementos hardware del ordenador.
  - **De reinicio** (inesperadas), del sistema, pulsar tecla reinicio…

- **De Software**, se producen como consecuencia directa de los procesos en ejecución.
  - **Llamadas al Sistema**: El usuario realiza una llamada al sistema (para hacer uso de un recurso del núcleo).

Interrupciones y excepciones

Interrupción de hardware o software

Cuando se produce una interrupción se pasa el control al sistema operativo, quien salva el contexto del proceso que se estaba ejecutando y se analiza la interrupción. Las interrupciones están catalogadas y el sistema operativo dispone de rutinas especiales para manipular cada tipo de interrupción. Una vez se ha atendido la interrupción la CPU continúa con su anterior tarea.



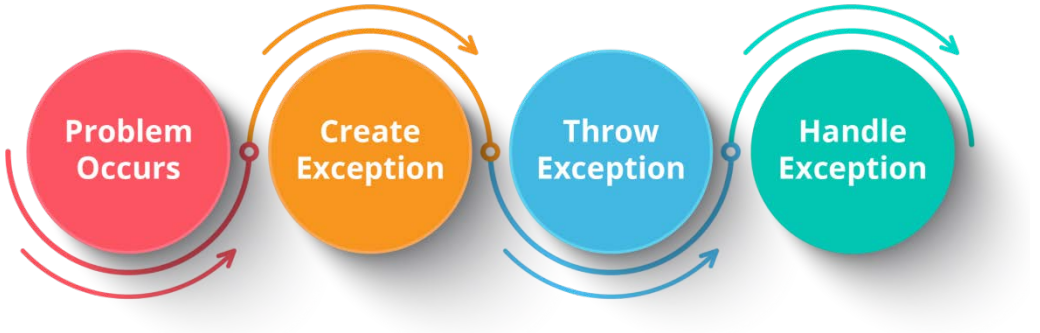
Interrupciones y excepciones

Excepción

Una **excepción** es un tipo de interrupción provocada por la propia CPU a causa de un error en la ejecución del proceso en activo como puede ser la realización de operaciones no permitidas, códigos de operación mal expresados, direcciones de memoria fuera de rango, etc.

Es el proceso o el proprio programa el que intenta llevar a cabo el manejo y control de dicho error.

El tratamiento de una excepción es similar al de la interrupción, con la salvedad de que las excepciones, no continúan el proceso con fallo sino que lo abortan.



Interrupciones y excepciones

Comparativa entre **interrupciones** y **excepciones**:

Interrupciones	Excepciones
Las interrupciones se presentan inesperadamente y sin relación con el proceso en ejecución. Son parte intrínseca del funcionamiento de cualquier sistema.	Las excepciones se producen como efecto directo de una instrucción concreta del proceso que se esta ejecutando. Aparecen por defectos de programación o errores graves.
El SO atienda la interrupción y a continuación continúa con la ejecución del proceso con la que estaba.	Son errores no recuperables.
Las interrupciones suelen tener asociados niveles de prioridad para su tratamiento.	Las excepciones no tienen asociados niveles de prioridad para su tratamiento.
Si se producen varias interrupciones simultáneamente, sólo se tratará una, quedando bloqueadas el resto.	Las excepciones se producen de una en una.