GESTIÓN DE SERVICIOS EN EL SISTEMA INFORMÁTICO

Capítulo 2.

Análisis de los procesos de sistemas

José Pablo Hernández

TEMARIO UD2.- Análisis de los procesos de sistemas

2.1. Identificación de procesos de negocio soportados por sistemas de información

2.2. Características fundamentales de los procesos electrónicos

2.2.1. Estados de un proceso

2.2.2. Manejo de señales, su administración y los cambios en las prioridades

2.3. Determinación de los sistemas de información que soportan los procesos de negocio y los activos y servicios utilizados por los mismos

2.4. Análisis de las funcionalidades de sistema operativo para la monitorización de los procesos y servicios

2.5. Técnicas utilizadas para la gestión del consumo de recursos

¿Qué es un proceso?

Proceso = Programa + Variables

Los programas necesitan pasar a la memoria para ejecutarse.

Un programa puede tener uno o varios procesos.

Un proceso puede invocar a ninguno, uno ó más procesos hijos.

¿Qué es un proceso?

Podemos decir que un programa es una entidad pasiva, en tanto en cuanto es un conjunto de instrucciones de código máquina y datos almacenados en un ejecutable.

Mientras que un proceso sería la ejecución de ese programa, es decir, el programa en acción.

Esto indica que los procesos son dinámicos, están en constante cambio debido a estos recursos necesarios, ya que al intentar realizar algún tipo de acción puede ser que tenga que permanecer a la espera de que dicho recurso esté disponible, por ejemplo una petición de lectura del disco duro, y que el brazo lector del disco duro lo esté utilizando otro proceso.

¿Qué es un proceso?

En los sistemas operativos multiproceso se intenta maximizar la utilización de la CPU, por lo que los procesos se ejecutan simultáneamente en la CPU y sólo quedan a la espera de ejecución si requieren de algún recurso del sistema que esté ocupado en ese momento, en cuanto obtiene dicho recurso podrá ejecutarse de nuevo.

Todo este proceso de gestión lo realiza el sistema operativo, por lo que es el que decide si un proceso es más prioritario que otros.

Es el programador el que decide esta prioridad, por ejemplo en el caso de Linux se utiliza un número en la programación de estrategias para garantizar la equidad de los procesos.

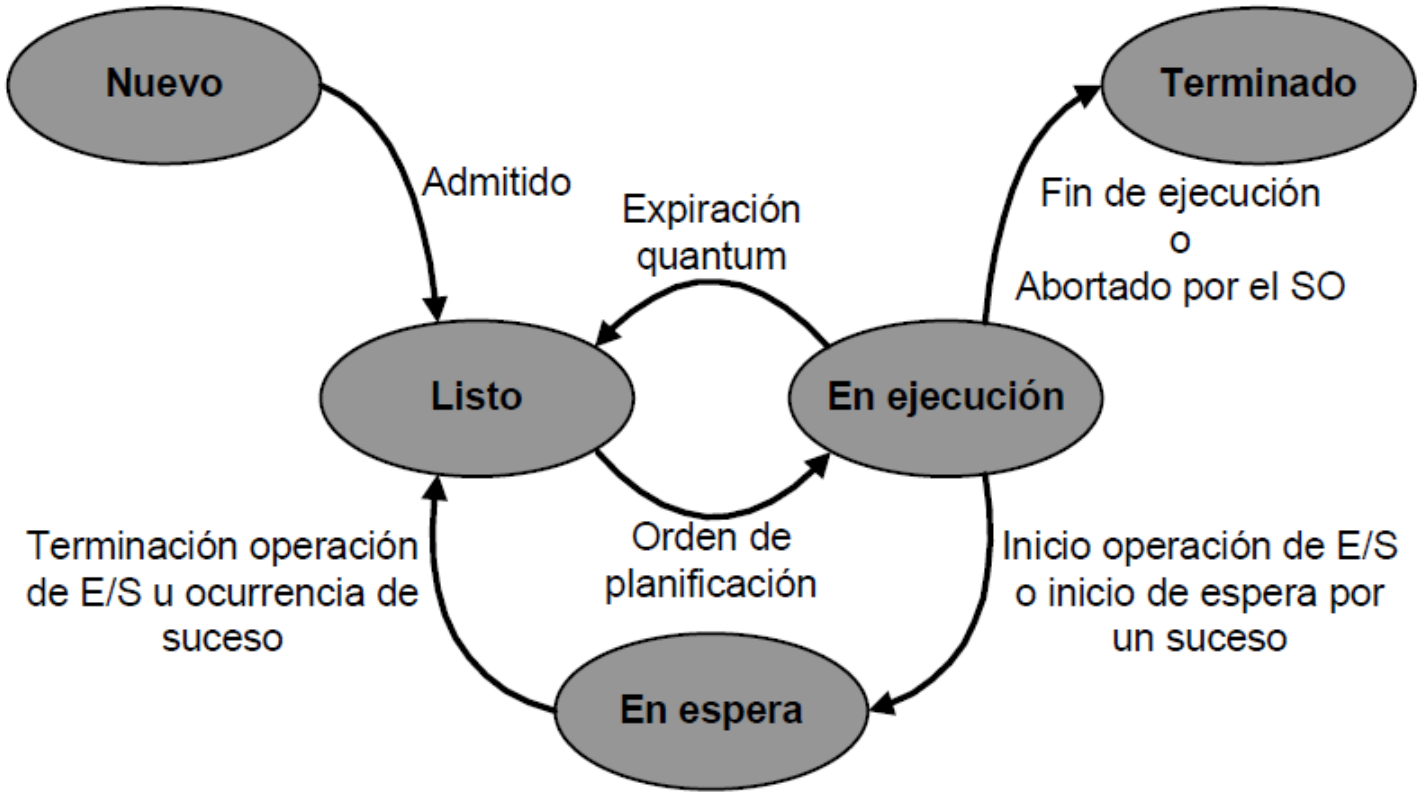
¿Qué es un proceso?

Cada proceso se representa en el sistema operativo con un bloque de control de proceso (PCB, Process Control Block).

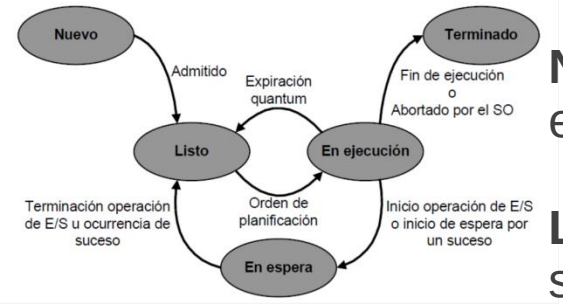
En este PCB se guardan una serie de elementos de información de los mismos.

Estos elementos son: el identificador del proceso, el estado del proceso, registros de CPU (acumuladores, punteros de la pila, registros índice y registros generales), información de planificación de CPU (prioridad de proceso, punteros a colas de planificación, etc.), información de gestión de memoria, información de contabilidad (tiempo de uso de CPU, números de procesos, etc.), información de estado de dispositivos E/S (lista de archivos abiertos, etc.).

Estados de un proceso



Estados de un proceso



Nuevo: El proceso se acaba de crear, pero aún no ha sido admitido en el grupo de procesos ejecutables por el sistema operativo.

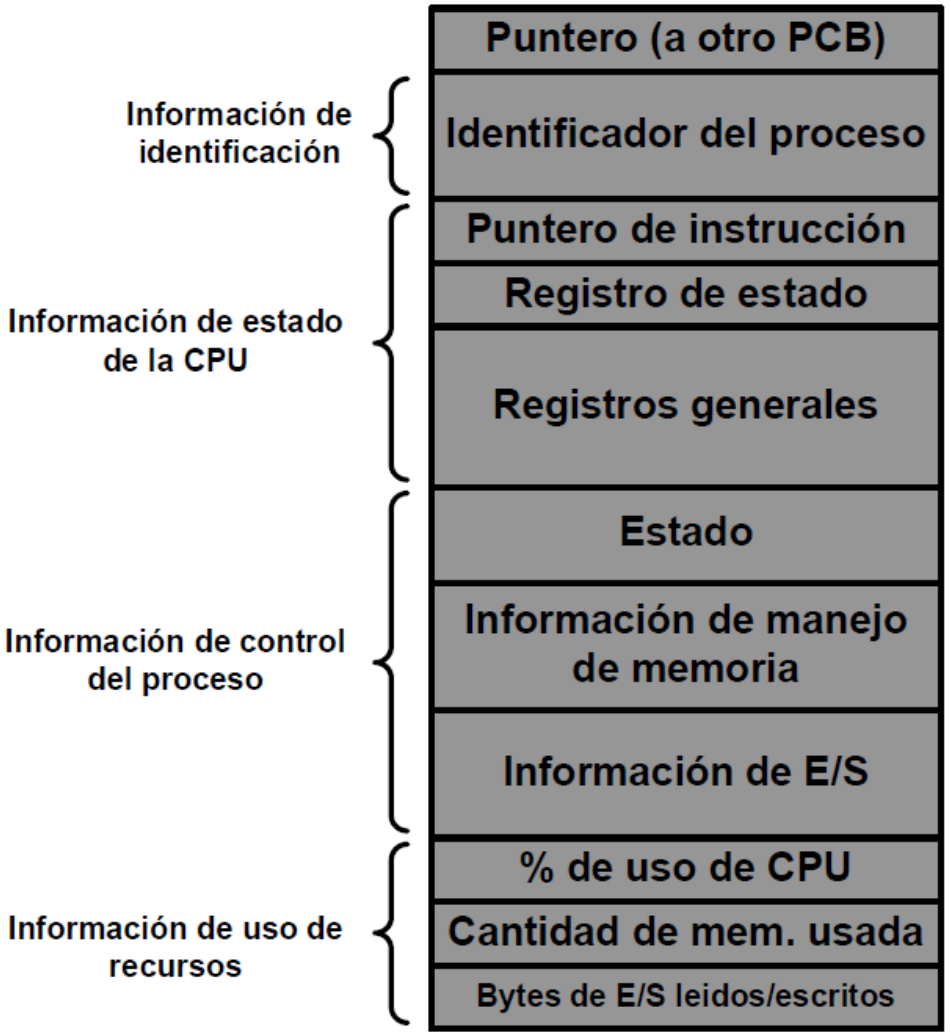
Listo: El proceso está esperando ser asignado al procesador para su ejecución.

En ejecución: El proceso tiene la CPU y ésta ejecuta sus instrucciones.

En espera: El proceso está esperando a que ocurra algún suceso, como por ejemplo la terminación de una operación de E/S.

Terminado: El proceso ha sido sacado del grupo de procesos ejecutables por el sistema operativo. Después de que un proceso es marcado como terminado se liberarán los recursos utilizados por ese proceso, por ejemplo, la memoria.

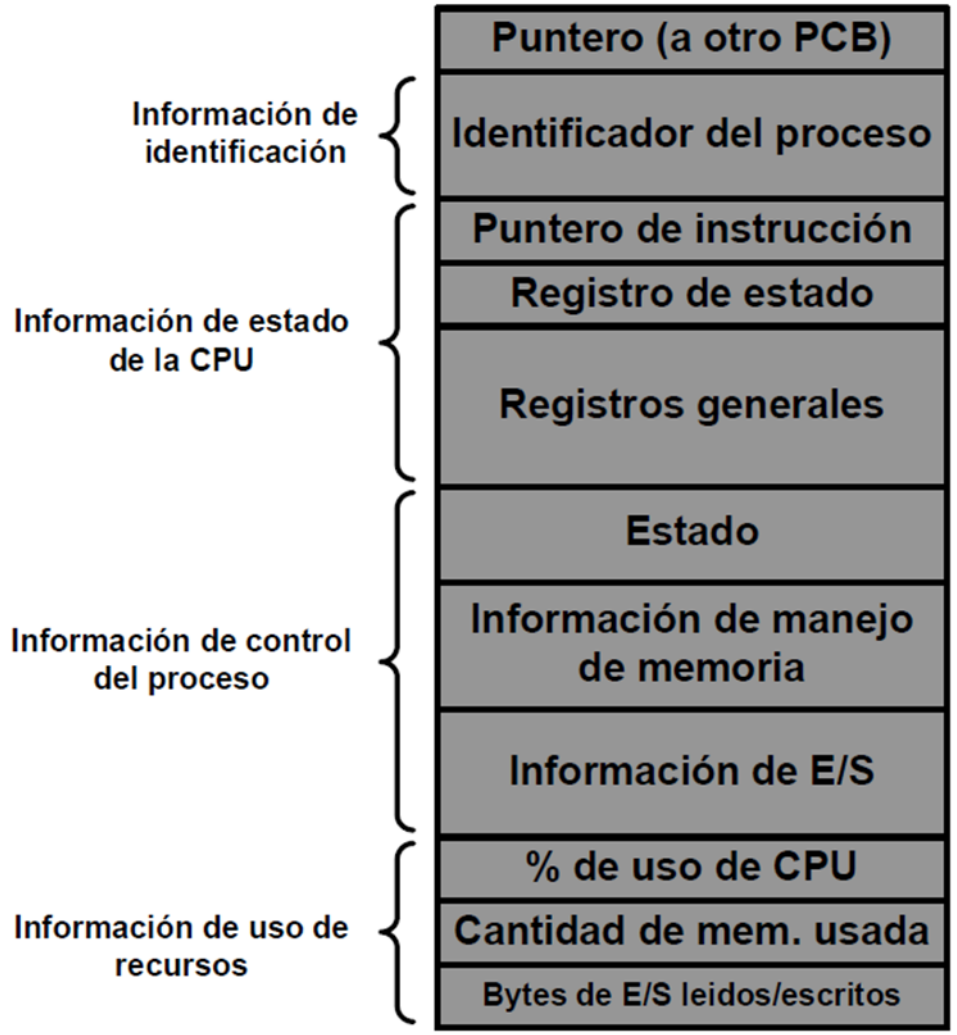
Bloque de control de proceso (PCB)



Un PCB es una estructura de datos que permite al sistema operativo controlar diferentes aspectos de la ejecución de un proceso.

El PCB se organiza en un conjunto de campos en los que se almacena información de diversos tipos.

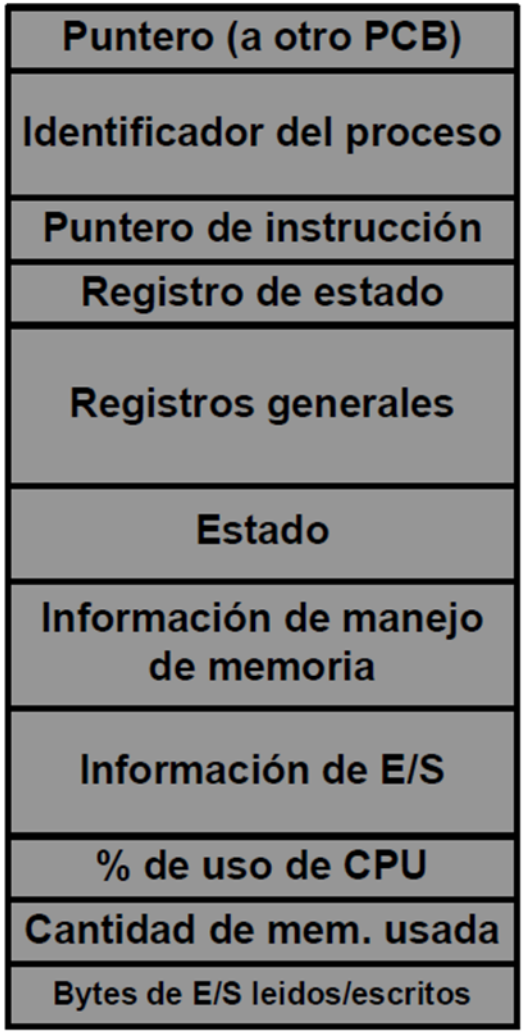
Los campos típicamente mantenidos en el PCB de un proceso se muestran en la figura.



Bloque de control de proceso (PCB)

· **Información de identificación**
Esta información está integrada básicamente por el **identificador del proceso** (PID), que es un número que identifica al proceso. Este número es diferente para todos los procesos que se encuentran en ejecución.

· **Información de estado de la CPU**
Se trata de un conjunto de campos que almacenan el estado de los registros de la CPU cuando el proceso es suspendido.



Bloque de control de proceso (PCB)

· **Información de control del proceso**
Se trata de un conjunto de información que es utilizada por el sistema operativo para controlar diversos aspectos de funcionamiento del proceso.
Pertenecen a esta categoría de información los siguientes campos:
– **Estado del proceso**: Listo, en ejecución, etc.
– **Información de manejo de memoria**: Como por ejemplo, la dirección física de memoria en la que se ubica la tabla de páginas del proceso.
– **Información de E/S**: Lista de ficheros abiertos, ventanas utilizadas, etc.

· **Información de uso de recursos**
Se trata de un conjunto de información relativa a la utilización realizada por el proceso de los recursos del sistema, como por ejemplo, el porcentaje de utilización de la CPU, la cantidad de memoria usada o los bytes de E/S escritos y leídos por el proceso.

Planificación de procesos

El objetivo de los sistemas multitarea es mantener múltiples programas en ejecución simultáneamente, pero como la CPU sólo puede ejecutar un programa de cada vez, hay que decidir quién se ejecuta en cada momento.

Se denomina planificación (*scheduling*) al mecanismo utilizado por el sistema operativo para determinar qué proceso (entre los presentes en el sistema) debe ejecutarse en cada momento.

Planificación de procesos.

Planificación en sistemas de tiempo compartido.

Los sistemas operativos más importantes del marcado actual (Windows, Linux, Mac OS y todas las versiones de Unix) se consideran sistemas operativos de tiempo compartido.

Objetivo prioritario de estos sistemas: Garantizar que el tiempo de respuesta de los programas se mantiene en unos valores admisibles para los usuarios.

Planificación de procesos.

Planificar los procesos es una tarea fundamental e importante, pues a través de ella se hacen referencia a un conjunto de políticas y mecanismos incorporados al Sistema Operativo que controlan el orden en que deben ser ejecutados los procesos, es por ello que el planificador se encarga de seleccionar el próximo proceso en hacer uso de la CPU, teniendo en cuenta que su objetivo fundamental es optimizar el rendimiento del Sistema Operativo.

El planificador, además de tener que escoger el proceso correcto para ejecutar, tiene que preocuparse también de realizar un uso eficiente de la CPU, debido a que la conmutación de procesos es muy costosa.

Planificación de procesos.

Planificación en sistemas de tiempo compartido.

Cuando un usuario interacciona con un programa y le da una orden, quiere que el programa responda en un tiempo razonable.

Para conseguir esto hay que hacer que el resto de programas que se encuentren en ejecución no monopolicen la CPU. Para ello, hay que ir repartiendo la CPU entre todos los programas, y además muy rápidamente, para que cada programa tenga una fracción del recurso CPU cada muy poco tiempo.

Planificación de procesos.

Planificación en sistemas de tiempo compartido.

Esquema de funcionamiento: A cada proceso en ejecución se le asigna un *quantum*, que representa el tiempo máximo que puede estar ocupando la CPU.

Entonces un proceso abandona la CPU, bien cuando se bloquea por una operación de E/S (pasando al estado “en espera”), o bien cuando expira su *quantum* (pasando al estado “listo”.)

Planificación de procesos.

¿Cuándo planificar?

Una cuestión clave relacionada con la planificación es en qué momento realizar las decisiones de planificación.

Resulta que hay una gran variedad de situaciones en las que es necesaria la planificación:

- **Cuando se crea un nuevo proceso.**
- **Cuando un proceso termina.**
- **Cuando un proceso se bloquea en E/S.**
- **Cuando tiene lugar una interrupción debida a una E/S**.

Planificación de procesos.

¿Cuándo planificar?

Cuando se crea un nuevo proceso:

Se necesita decidir si se ejecuta el proceso padre o el proceso hijo. Ya que ambos procesos están en el estado de listos, se trata de una decisión de planificación normal y puede hacerse de cualquiera de las dos maneras, esto es, el planificador puede legítimamente elegir ejecutar a continuación bien el padre, o bien el hijo.

Planificación de procesos.

¿Cuándo planificar?

Cuando un proceso termina:

Ya que no puede seguirse ejecutando ese proceso (puesto que ya no existe), debe elegirse a algún otro proceso de entre el conjunto de procesos listos. Si no hay ningún proceso preparado, normalmente se ejecuta un proceso ocioso proporcionado por el sistema.

Planificación de procesos.

¿Cuándo planificar?

Cuando un proceso se bloquea en E/S:

A veces el motivo del bloqueo puede jugar algún papel en la elección. Por ejemplo, si *A* es un proceso importante y está esperando a que *B* salga de su región crítica, el dejar que *B* se ejecute a continuación puede permitir que salga de su región crítica, haciendo posible que *A* pueda continuar. Sin embargo, el problema es que generalmente el planificador no cuenta con la información necesaria para poder tomar en cuenta esta dependencia entre los procesos.

Planificación de procesos.

¿Cuándo planificar?

Cuando tiene lugar una interrupción debida a una E/S:

Si la interrupción proviene de un dispositivo de E/S que ha completado ahora su trabajo, entonces algún proceso que estaba bloqueado esperando por la E/S puede ahora estar preparado para ejecutarse. Corresponde al planificador decidir si debe ejecutarse el nuevo proceso preparado, si debe continuar ejecutándose el proceso en ejecución en el momento de la interrupción, o si debe ejecutarse algún tercer proceso.

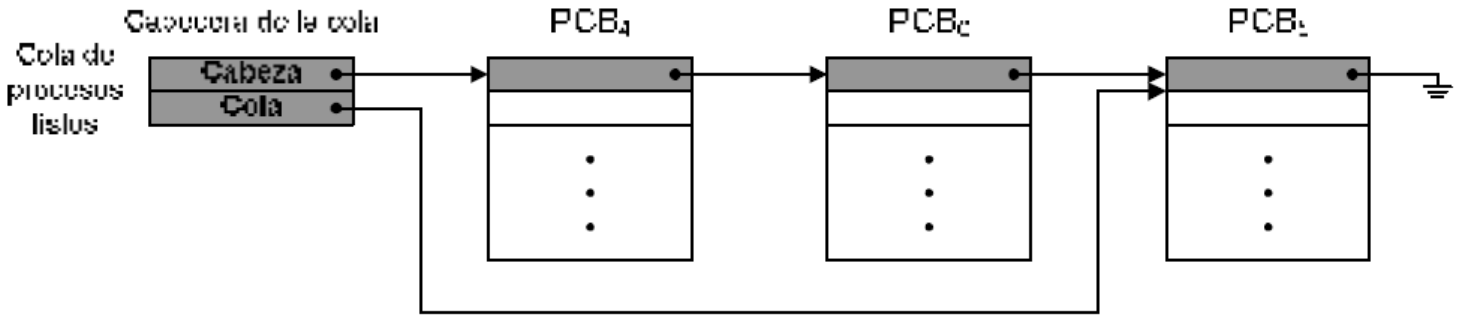
Planificación de procesos.

Colas de planificación.

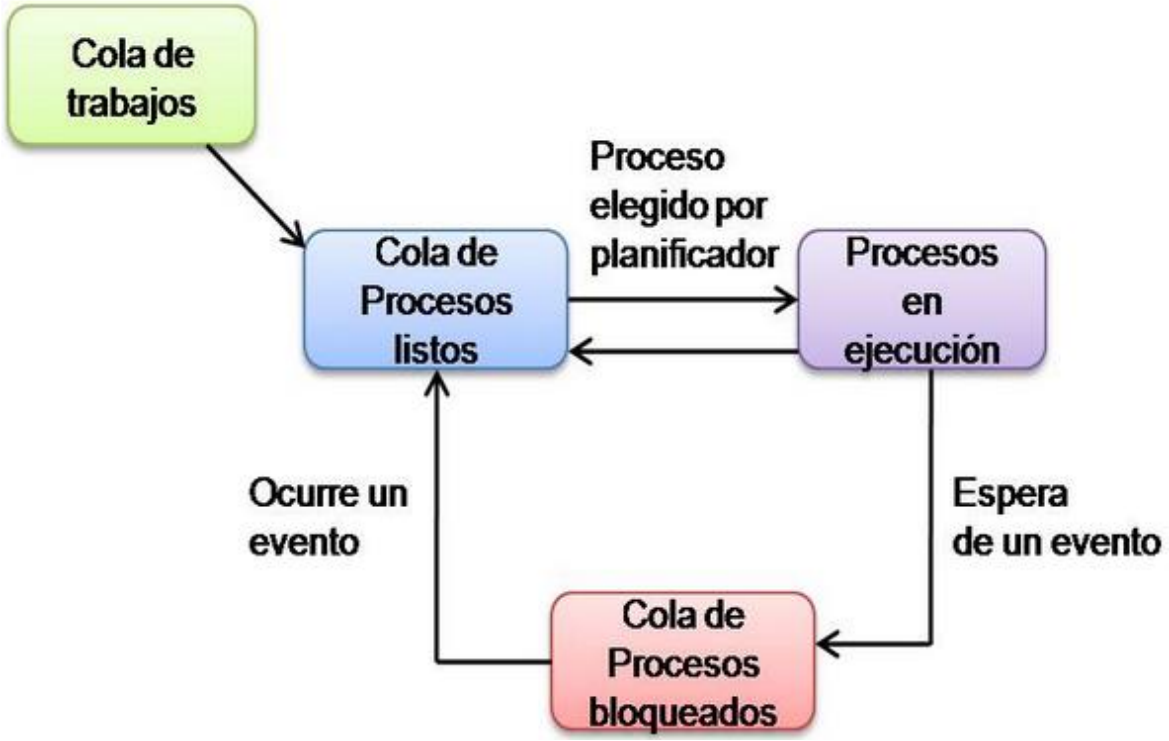
Son unas estructuras de datos que organizan los PCBs de los procesos que se encuentran cargados en el sistema en función de su estado.

El SO planifica los procesos en función de la información mantenida en estas colas.

Estas estructuras se forman enlazando los PCBs de los procesos mediante punteros.



Planificación de procesos.



Colas de planificación.

Existen dos tipos de colas (principalmente):

· **Cola de procesos listos**: Contiene a los procesos que se encuentran en el estado “listo”.

· **Cola de dispositivo**: Contiene los procesos que están esperando por un determinado dispositivo. Estos procesos se encuentran en el estado “En espera”. Cada dispositivo tiene una cola asignada.

Planificación de procesos.

Políticas de planificación.

Se pueden definir múltiples políticas de planificación de procesos: por orden de llegada, primero la tarea más breve, por orden de prioridad, etc.

En definitiva, lo que una política de planificación debe conseguir es que los procesos obtengan adecuadamente sus turnos de ejecución por lo que son tratados de la misma forma, que no se produzca sobrecarga, es decir, el planificador debe responder rápidamente ante cargas de trabajo ligera y responder de la misma forma ante cargas de trabajo similares.

Y obtener un buen rendimiento, por lo se debe lograr finalizar el mayor número de procesos y maximizar el tiempo de respuesta.

Planificación de procesos.

Políticas de planificación.

No existe una política de planificación óptima para todas las computadoras, sino que depende de las características de los procesos.

Así se puede ver cómo una política obtiene unos resultados excelentes en un sistema, sin embargo en otro sistema el rendimiento es mucho menor.

Ello se debe a las características de los procesos, donde cada uno puede tener una cantidad de operaciones de E/S enorme cómo es el caso de las bases de datos, otros usan mayormente la CPU, otros realizan una mayor lectura de datos frente a otros, hay procesos que requieren una prioridad máxima en los turnos de ejecución, es el caso de los procesos de tiempo real, y hay procesos que requieren más tiempo de ejecución que otros, por lo que habrá que valorar si terminar primero los cortos o no.

Planificación de procesos.

Políticas de planificación. Políticas más comunes. FCFS.

Las siglas **FCFS** significan en inglés First Come First Served (Primero en llegar, Primero en ser Servido), dentro de los algoritmos de Planificación de la CPU, este es el más sencillo.

La carga de trabajo se procesa simplemente en un orden de llegada. Por no tener en consideración el estado del sistema ni las necesidades de recursos de los procesos individuales, la planificación FCFS puede dar lugar a pobres rendimientos.

Planificación de procesos.

Políticas de planificación. Políticas más comunes. FCFS.

Este algoritmo posee un alto tiempo de respuesta de la CPU pues el proceso no abandona la CPU hasta no haber concluido pues es un algoritmo Sin Desalojo.

La planificación FCFS elimina la noción e importancia de las prioridades de los procesos.

Para elegir el proceso al cual se le asignará la CPU, se escoge el que lleva más tiempo listo (primero en la cola).

Planificación de procesos.

Políticas de planificación. Políticas más comunes. FCFS.

El proceso que se está ejecutando solo cede la CPU por dos razones:

- Que se bloquee voluntariamente en espera de un evento. (Impresora, fichero, etc.)
- Cuando termina su ejecución.

La ventaja de este algoritmo es su fácil implementación, sin embargo, no es válido para entornos interactivos ya que un proceso de mucho cálculo de CPU hace aumentar el tiempo de espera de los demás procesos.

Planificación de procesos.

[FCFS]

Desventajas

Ventajas

Algunas de las características de este algoritmo es que es **no apropiativo** y justo en el sentido formal, aunque injusto en el sentido de que: **los trabajos largos hacen esperar a los cortos** y los **trabajos sin importancia hacen esperar a los importantes**.

Por otro lado es predecible, pero no garantiza buenos tiempos de respuesta, por ello se emplea como esquema secundario.

[◄] [►]

Planificación de procesos.

Políticas de planificación. Políticas más comunes. SJF.

El algoritmo SJF (Shortest-Job-First) se basa en los ciclos de vida de los procesos, los cuales transcurren en dos etapas o periodos que son: ciclos de CPU y ciclos de entrada/salida, también conocidos por ráfagas.

La palabra shortest (el más corto) se refiere al proceso que tenga el próximo ciclo de CPU mas corto.

Planificación de procesos.

Políticas de planificación. Políticas más comunes. SJF.

La idea es escoger entre todos los procesos listos el que tenga su próximo ciclo de CPU más pequeño. El SJF se puede comportar de dos formas:

- Con Desalojo: Si se incorpora un nuevo proceso a la cola de listos y este tiene un ciclo de CPU menor que el ciclo de CPU del proceso que se está ejecutando, entonces dicho proceso es desalojado y el nuevo proceso toma la CPU.
- Sin desalojo: Cuando un proceso toma la CPU, ningún otro proceso podrá apropiarse de ella hasta que el proceso que la posee termine de ejecutarse.

.

Planificación de procesos.

Políticas de planificación. Políticas más comunes. SJF.

El SJF se considera como un algoritmo óptimo, porque da el mínimo tiempo de espera promedio para un conjunto de procesos, así como las estimaciones de CPU.

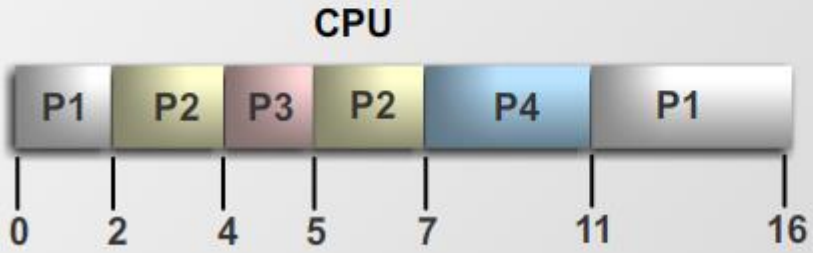
Su dificultad radica en que materialmente es un algoritmo imposible de implementar.

Planificación de procesos.

[SJF]

Ejemplo de SJF Apropiativo

Proceso	Tiempo de Llegada	Tiempo de Ejecución en CPU (te)
P1 7	0.0	7
P2 4	2.0	4
P3 1	4.0	1
P4 4	5.0	4



P1 al llegar a la cola entra a la CPU (0.0). Pero a los 2 tiempos de estarse ejecutando P1 llega P2 y como P2 (4 te) es menor que el tiempo remanente de P1 (5 te) entonces P2 se apropia de la CPU, y P1 va a la cola de listos. Luego a los 2 tiempos más, llega P3 (1 te) que es menor que el tiempo remante de P2, P3 toma la CPU y P2 va a la cola. Al concluir P1 regresa P2 (2te), luego al terminar comienza P4 (4 te) y por último regresa P1 (5 te) Asi es como se reorganiza la cola y la CPU en el esquema Apropiativo.

[◄] [►]

Planificación de procesos.

Políticas de planificación. Políticas más comunes. R-R.

El Round Robin es uno de los algoritmos más antiguos, sencillos y equitativos en el reparto de la CPU entre los procesos lo que significa que evita la monopolización de uso de la CPU, y es muy válido para entornos de tiempo compartido.

El algoritmo consiste en definir una unidad de tiempo pequeña, llamada “quantum” o “cuanto” de tiempo, la cual es asignada a cada proceso que esté en estado listo.

Si el proceso agota su quantum (Q) de tiempo, se elige a otro proceso para ocupar la CPU. Si el proceso se bloquea o termina antes de agotar su quantum también se alterna el uso de la CPU.

Planificación de procesos.

Políticas de planificación. Políticas más comunes. R-R.

Es por ello que surge la necesidad de un reloj en el sistema.

El reloj es un dispositivo que genera periódicamente interrupciones.

Esto es muy importante, pues garantiza que el sistema operativo (en concreto la rutina de servicio de interrupción del reloj) coja el mando de la CPU periódicamente.

El quantum de un proceso equivale a un número fijo de pulsos o ciclos de reloj.

Al ocurrir una interrupción de reloj que coincide con la finalización del quantum se llama al despachador, el cual le cede el control de la CPU al proceso seleccionado por el planificador.

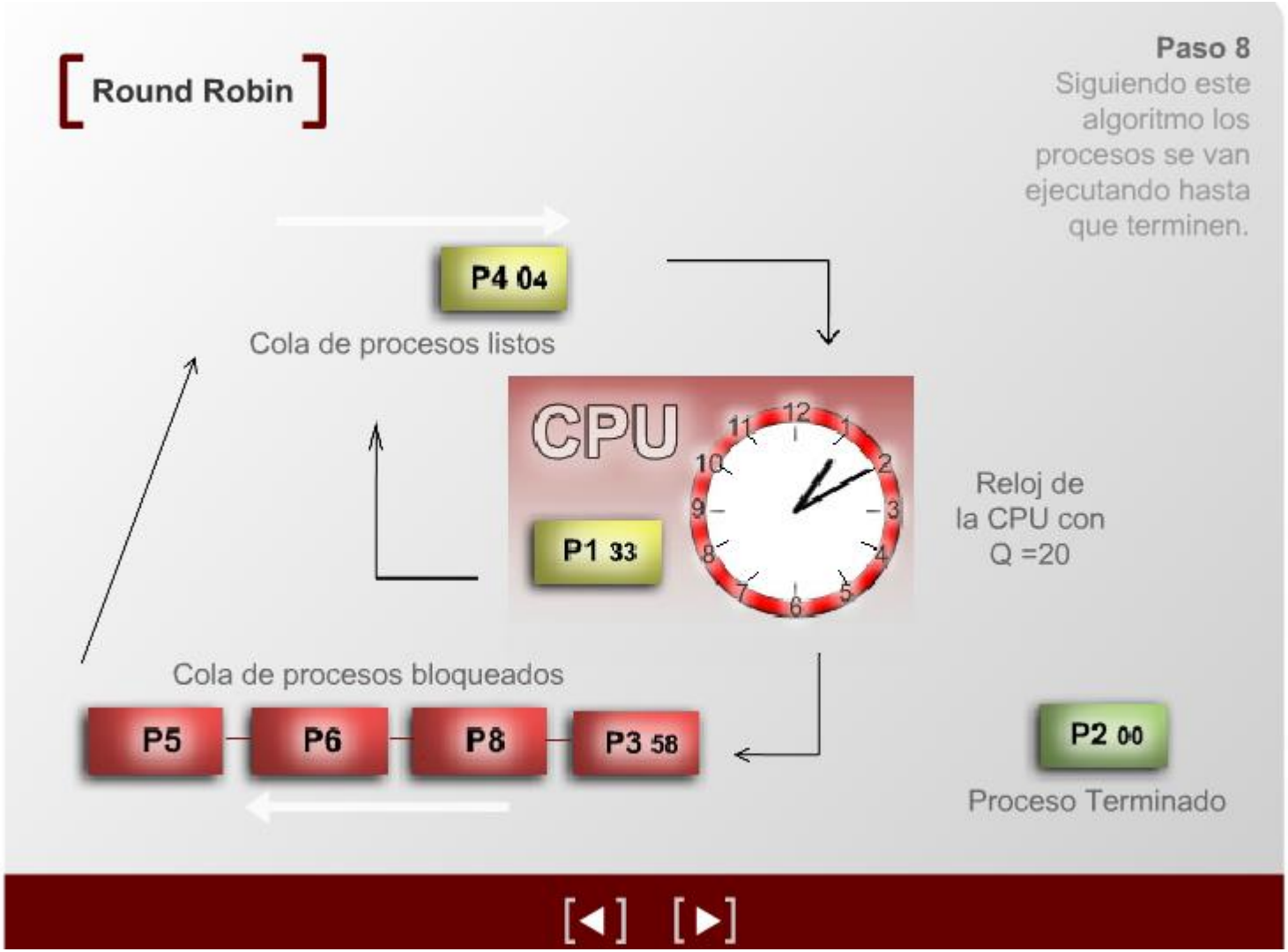
Planificación de procesos.

Políticas de planificación. Políticas más comunes. R-R.

Un proceso puede abandonar la CPU por 2 criterios:

- Libremente, si su tiempo de ejecución en la CPU es < Q (quantum).
- Después de una interrupción, si su tiempo de ejecución en la CPU es > Q (quantum) o si el proceso se bloquea.

Planificación de procesos.



Planificación de procesos.

Políticas de planificación.

Los procesos en Linux pueden ser divididos en tres categorías: interactivos, tiempo real o por lotes.

Los procesos de tiempo real son manejados bien por un algoritmo FIFO o turno rotatorio (Round-Robin), ya que al ser procesos considerados como prioritarios deben de ser ejecutados antes que los demás.

Los demás procesos son manejados utilizando una planificación Round-Robin con un sistema de envejecimiento utilizado en la planificación de prioridades mencionada.

Planificación de procesos.

Políticas de planificación.

En Windows, la planificación de procesos se basa en la utilización de colas múltiples de prioridades.

Posee 23 niveles de colas clasificadas de la 31-16 en clase de tiempo real y las demás en clase variable.

Cada cola es manejada mediante Round-Robin, pero si llega un proceso con mayor prioridad, se le es asignado el procesador.

Planificación de procesos.

Políticas de planificación.

El caso de Mac OS X es parecido al de Windows, utilizando varias colas de procesos cada una con un nivel de prioridad.

Un hilo puede pasar de una cola a otra dependiendo de los requerimientos.

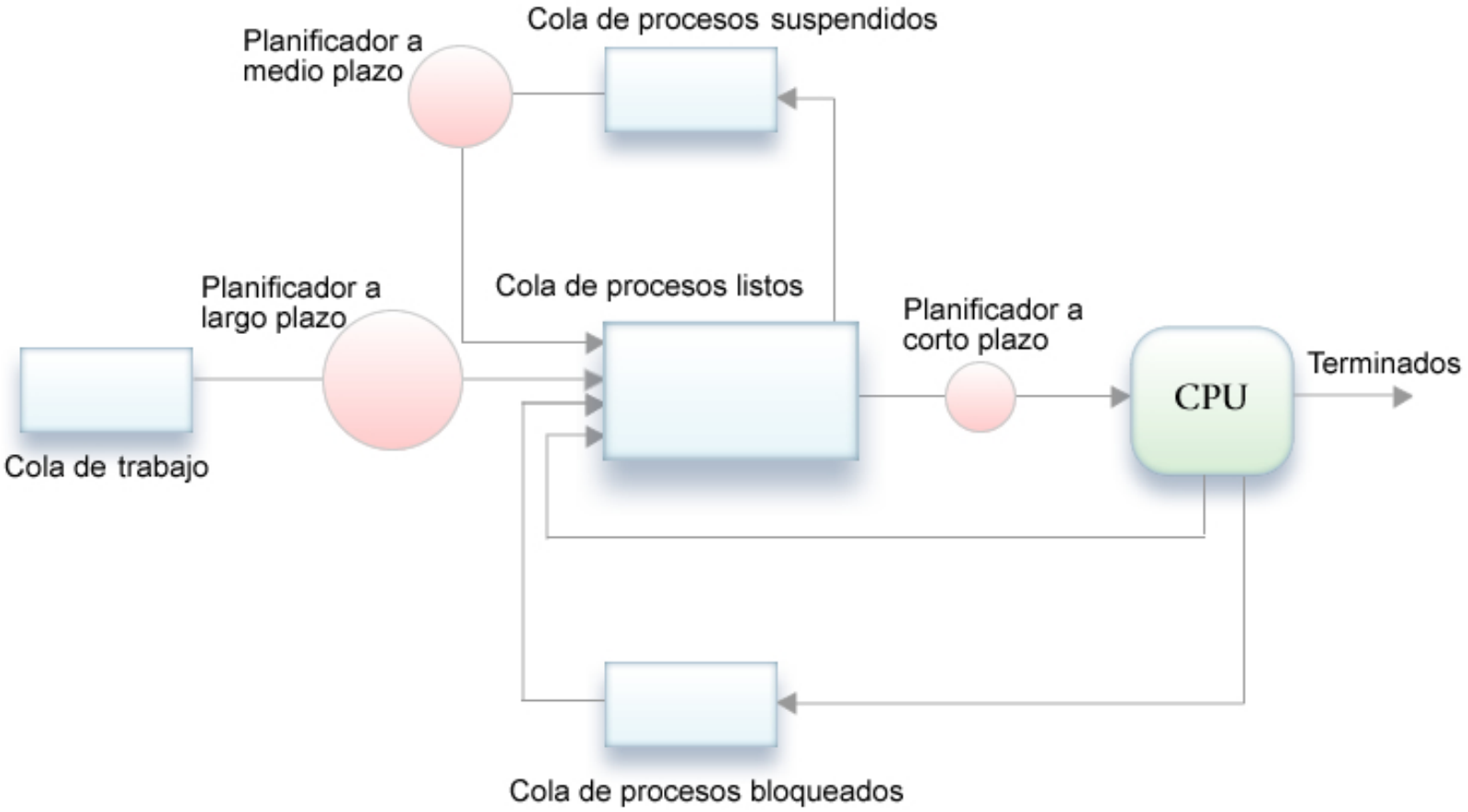
Estos niveles se pueden manejar mediante llamadas al sistema.

Los niveles son: normal, alta, Kernel y tiempo real.

Planificación de procesos.

Planificadores.

Ejemplo de diferentes planificadores:



Planificación de procesos.

Planificadores.

Existen diferentes planificadores en el sistema.

Primero nos encontramos el **planificador a largo plazo**, el cual es el encargado de controlar el grado de multiprogramación en el sistema, intentando conseguir una mezcla adecuada de trabajos en CPU y E/S.

Es por tanto el encargado de suministrar los procesos a la cola de planificación a corto plazo.

Planificación de procesos.

Planificadores.

Existen diferentes planificadores en el sistema.

Existe también un **planificador a medio plazo.** Es el encargado de suspender y posteriormente restaurar procesos de poco interés, realizando el intercambio de los mismos entre la memoria principal y el disco o memoria secundaria.

Dicho proceso es conocido como *swapping*, y se ejecuta cuando hay escasez de recursos.

Planificación de procesos.

Planificadores.

Existen diferentes planificadores en el sistema.

El **planificador a corto a plazo** es el encargado de asignar y desasignar la CPU.

Su trabajo es coger un proceso de la cola de procesos preparados y asignarle la CPU.

Hay dos tipos de planificadores a corto plazo:

- No expulsivas, el proceso abandona la CPU cuando termina o a la espera de un suceso externo.

- Expulsivas, el proceso que se está ejecutando puede pasar a estado listo enviado por parte del sistema operativo, es el caso de los sistemas de tiempo compartido y tiempo real, como UNIX, Windows NT/XP o superior, MAC OS X.

Planificación de procesos.

Operaciones sobre procesos.

Los procesos tienen que poder ser creados y eliminados dinámicamente en el sistema.

Debido a ello, el sistema debe proporcionar facilidades para llevar a cabo estas acciones con los procesos.

Las funcionalidades básicas se indican a continuación.

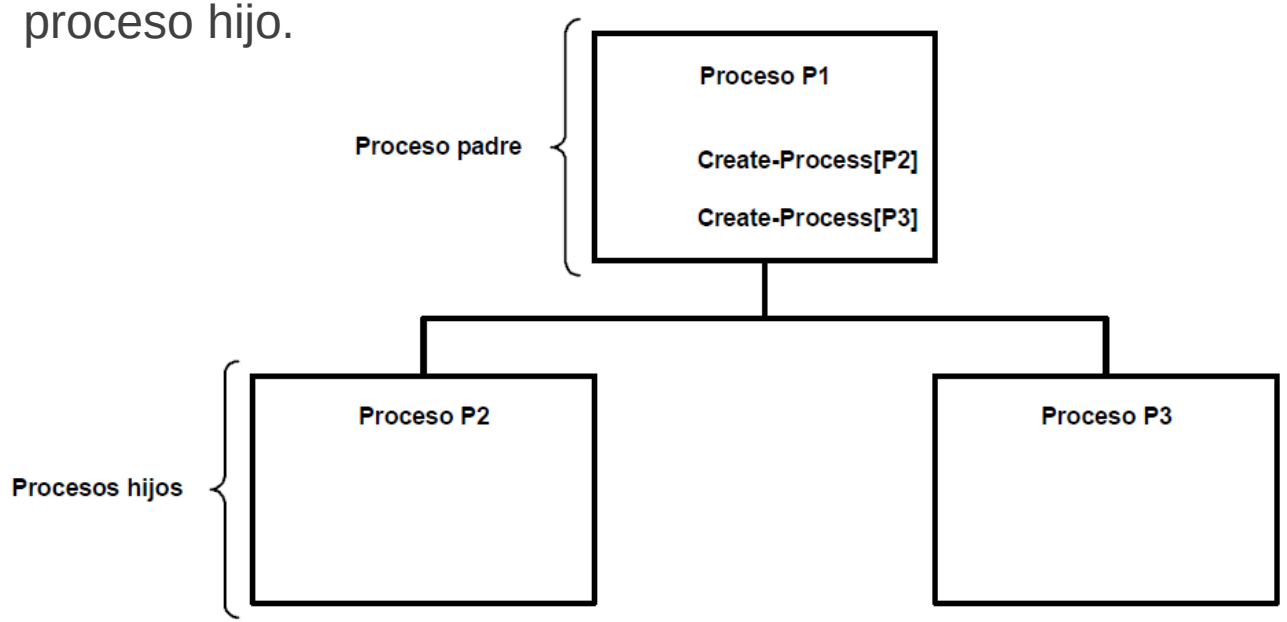
Planificación de procesos.

Operaciones sobre procesos.

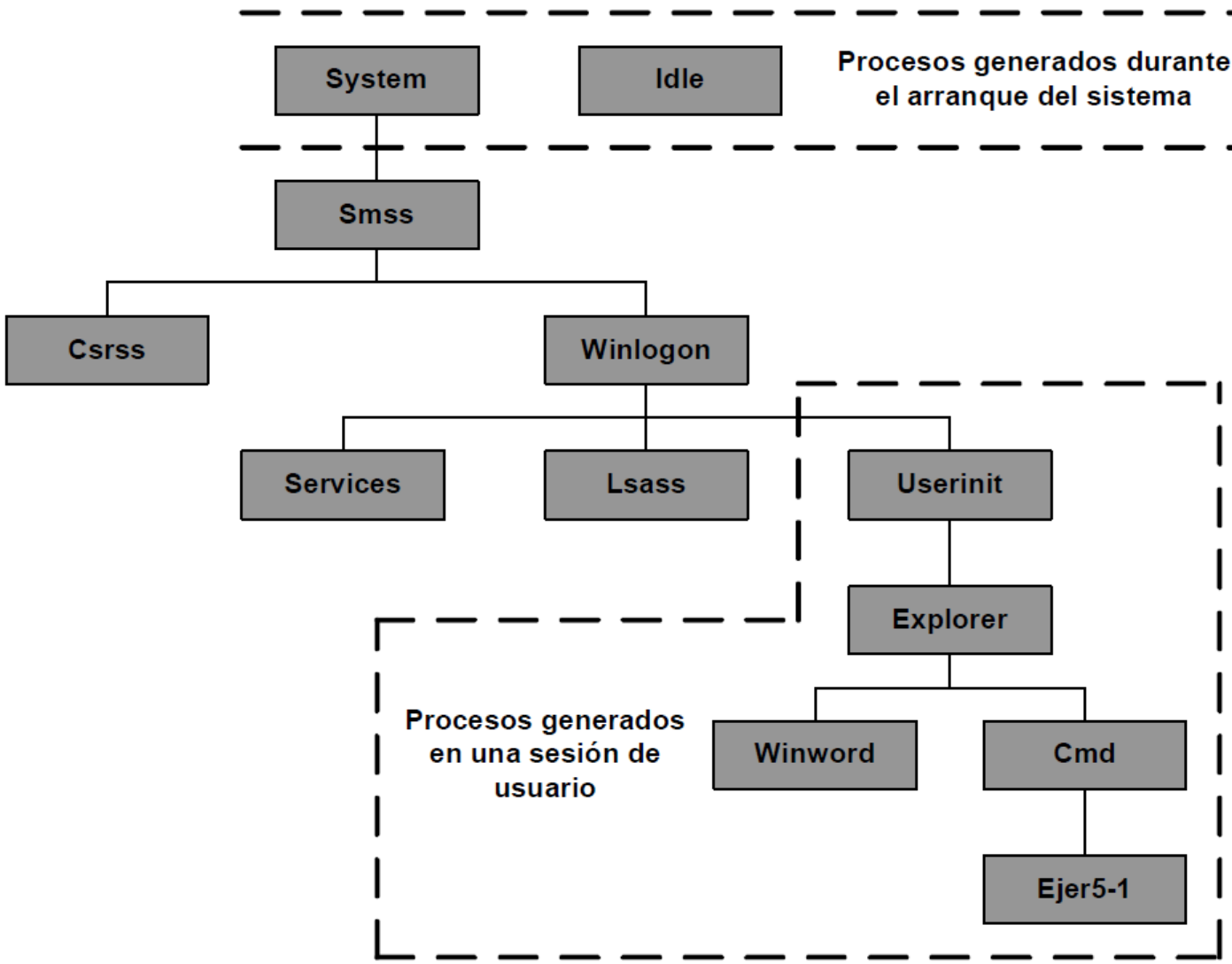
Creación de procesos

Todo sistema operativo debe proporcionar un servicio Create-Process, que será utilizado por un proceso para crear otro proceso.

Al proceso que solicita el servicio Create-Process se le denomina proceso padre, y al proceso que es creado mediante este servicio, proceso hijo.



Planificación de procesos.



Árbol de procesos típico de Windows

Siempre debe haber un proceso raíz, que será creado durante el arranque del sistema. En el caso de Windows este proceso se llama System.

También se crea durante el arranque el proceso idle, *que es un proceso que no hace nada. Este proceso es el que se ejecuta cuando la CPU no tiene ningún otro proceso para ejecutar.*

Planificación de procesos.

Operaciones sobre procesos.

Terminación de procesos

Un proceso puede terminar por sí mismo, o bien puede ser terminado por otro proceso, que generalmente sólo puede ser su proceso padre.

Un proceso termina por sí mismo llamando a un servicio del sistema, denominado normalmente Exit o Exit-Process.

Un proceso puede terminar la ejecución de un proceso hijo llamando a un servicio del sistema, conocido normalmente como Abort o Terminate-Process.

El que un proceso haga que termine otro proceso es una situación extraordinaria, normalmente ligada a la ocurrencia de errores. Cuando las cosas van bien, los procesos terminan por sí mismos.

Planificación de procesos.

Operaciones sobre procesos.

UNIX

Los procesos en los sistemas UNIX están identificados por un número que es único (PID), además de que cada proceso el espacio de memoria utilizado, formado por tres segmentos: el código, los datos y la pila.

También contiene la información de control del proceso, que indica la planificación y estado del proceso, la estructuración de datos y la comunicación entre procesos.

Otra información importante es el número de identificación de usuario (UID) y el número de identificación del grupo de usuarios al que pertenece el proceso (GID).

Todo ello pertenece al bloque de control de proceso (PCB).

Planificación de procesos.

Operaciones sobre procesos.

UNIX

La creación y destrucción de procesos en UNIX se ajusta a la filosofía de la manera más sencilla posible, así las llamadas al sistema tienen el mínimo número de parámetros.

Las llamadas correspondientes a la creación, destrucción y bloqueo espera de un proceso son respectivamente:

- Fork
- Exit
- Wait

Planificación de procesos.

Operaciones sobre procesos.

UNIX

La llamada fork crea un nuevo proceso hijo idéntico al proceso padre.

Tienen la misma imagen de memoria, el mismo bloque de control de proceso y los mismos archivos abiertos, aunque situados en distintos espacios de memoria.

Para poder distinguir a ambos procesos, la llamada fork devuelve distintos valores, el hijo recibe un valor 0 y el padre el PID del hijo.

Planificación de procesos.

Operaciones sobre procesos.

UNIX

La llamada exit finaliza un proceso.

Cuando se produce esta finalización del proceso hijo, se manda al sistema la llamada wait, para que el padre se bloquee a la espera de la finalización del hijo.

Si el proceso hijo finalizara antes de que el padre recibiera esta llamada, el proceso hijo se convertiría en un proceso en estado zombie, y hasta que no se ejecute esta llamada wait el proceso no se eliminará.

Para evitar la acumulación de procesos UNIX prevé un límite de números de procesos zombie y aquellos procesos hijos que se destruyen más tarde que sus procesos padres, al quedar huérfanos sería el primer proceso del sistema (init) el que se encarga de recoger su estado de finalización.

Planificación de procesos.

Operaciones sobre procesos.

UNIX

La llamada wait bloquea el proceso que lo ha llamado hasta que uno de sus procesos hijos es destruido, por lo que si el proceso no tiene hijos wait regresa y el valor devuelto es igual al Pid de dicho proceso hijo.

Planificación de procesos.

Operaciones sobre procesos.

UNIX

En UNIX se puede utilizar un comando para eliminar un proceso que se está ejecutando por la razón que estimemos oportuno; gran consumo de recursos, ya no es necesario, etc.

Esta orden es “kill *Pid proceso a eliminar*”, por lo que para eliminar un proceso primero se debe conocer su Pid.

Planificación de procesos.

Operaciones sobre procesos.

Windows

Un programa en Windows es controlado por eventos.

Así el programa principal espera la llegada de un evento como puede ser al presionar una tecla, y posteriormente invoca un procedimiento para procesar dicho evento, actualización de pantalla, del programa, etc.

Planificación de procesos.

Operaciones sobre procesos.

Windows

Windows también tiene llamadas al sistema al igual que UNIX, de hecho el número de llamadas es extremadamente grande.

En Windows encontramos que por cada llamada al sistema existe un procedimiento de biblioteca que los invoca.

Por ello Windows ha creado un conjunto de procedimientos, llamado API Win32, que se ha de utilizar para solicitar servicios al sistema operativo.

Planificación de procesos.

Operaciones sobre procesos.

Windows

La creación de procesos en Windows se genera mediante la llamada CreateProcess, que tanto crea el proceso como carga el programa en el nuevo proceso.

Esta llamada tiene 10 parámetros: el programa a ejecutar, atributos de seguridad, bits de control de archivos abiertos, prioridad, especificación de la ventana a crear y un apuntador a la estructura a la que al invocador se le devuelve información del proceso recién creado.

CreateProcess tiene 100 funciones más para administrar y sincronizar procesos.

Planificación de procesos.

Operaciones sobre procesos.

Windows

Al igual que en UNIX, en Windows se crea un nuevo proceso hijo a partir del padre, el cual tiene su propio espacio de direcciones como en UNIX, pero mientras en UNIX el espacio de direcciones del hijo era una copia del padre, en Windows el espacio de direcciones del hijo es completamente diferente al del padre desde el principio.

Planificación de procesos.

Operaciones sobre procesos.

Windows

La llamada en Windows relativa a la destrucción de un proceso es ExitProcess, y al igual que ocurría con UNIX cuando se especifica la llamada WaitForSingleObject junto con un parámetro que especifica un proceso, quien lo invoca espera hasta que se produce la finalización del proceso.

Planificación de procesos.

Operaciones sobre procesos.

Windows

En Windows encontramos multitud de llamadas al sistema igual que en UNIX, aunque hay algunas en UNIX que no se pueden utilizar en Windows, como es el manejo de enlaces, manejo de montaje de unidades, etc. Si en UNIX utilizamos la orden kill Pid para eliminar un proceso, en Windows existe la función TerminateProcess.

Planificación de procesos.

UNIX	WIN32	Descripción
Fork	CreateProcess	Crea un proceso nuevo
Waitpid	WaitForSingleObject	Puede esperar a que un proceso termine
Execve	(ninguna)	CreateProcess= fork + execve
Exit	ExitProcess	Termina la ejecución
Open	CreateFile	Crea un archivo o abre uno existente
Close	CloseHandle	Cierra un archivo
Read	ReadFile	Lee datos de un archivo
Write	WriteFile	Escribe datos en un archivo
Lseek	SetFilePointer	Mueve el apuntador de archivo
Stat	GetFileAttributesEx	Obtiene diversos atributos de archivo
Mkdir	CreateDirectory	Crea un directorio nuevo
Rmdir	RemoveDirectory	Elimina un directorio vacío
Link	(ninguna)	Win32 no maneja enlaces
Mount	(ninguna)	Win32 no maneja montajes
unmount	(ninguna)	Win32 no maneja montajes
Chdir	SetCurrentDirectory	Cambio el directorio de trabajo actual
Kill	TerminateProcess	Elimina un proceso
Time	GetLocalTime	Obtiene la hora actual

Planificación de procesos.

Dispatcher.

Una vez que el planificador ejecuta y elige el proceso a asignar al procesador, se invoca al despachador (dispatcher) que es el encargado de asignar el proceso al procesador.

Es decir, el planificador elige el proceso que entra a la CPU, en ese momento el despachador toma el proceso que estaba en la CPU y actualiza el PCB del proceso en su CPU virtual, luego elige el proceso seleccionado por el planificador y le asigna la CPU.

Asignar a la CPU no es más que cargar la CPU virtual del proceso y reemplazarlo en la CPU.

La tarea que realiza es:

- Cambiar el contexto. Salvar registros del procesador en PCB del proceso saliente. Cargar los registros con los datos del PCB del proceso entrante.
- Saltar a la instrucción adecuada que había quedado el proceso que se asigno a la CPU (registro program counter).

Cooperación entre procesos

En la mayoría de las ocasiones los procesos son entidades totalmente aisladas: llevan a cabo su trabajo sin tener que comunicarse con otros procesos o programas.

Son programas muy simples que no necesitan comunicarse con otros programas. Sin embargo las cosas en la realidad no son tan sencillas.

En muchas ocasiones, los programas o procesos necesitan intercambiar información entre sí.

Pongamos dos ejemplos:

- En una plataforma Windows, el intercambio de información a través del portapapeles.
- Chatear a través de la red. Hay dos procesos (dos navegadores) que intercambian información.

Son dos casos totalmente diferentes, pero son dos ejemplos claros de programas que cooperan entre sí.

Cooperación entre procesos

La cooperación entre procesos requiere que estos se comuniquen. A continuación se indican los mecanismos básicos de comunicación:

- **Memoria compartida**:
  - Se basa en que los procesos que desean comunicarse compartan una misma región de memoria física. Para llevar a cabo la comunicación, uno escribe y otro lee de la región de memoria compartida.
  - Los procesos utilizan servicios del sistema operativo para compartir la región.
- **Paso de mensajes**:
  - Los procesos utilizan una pareja de servicios del sistema operativo para comunicarse. Estos servicios son conocidos habitualmente como *Send* y *Receive*.
  - Para llevar a cabo la comunicación un proceso ejecuta la función *Send* y el otro *Receive*, intercambiando de esta forma un bloque de información que recibe el nombre de mensaje.