GENERALITAT VALENCIANA

Conselleria d'Educació, Investigació, Cultura i Esport

TEMA 1.1.1

ACTIVIDADES - PLANIFICACIÓN PROCESOS

PROGRAMACIÓN DE SERVICIOS Y PROCESOS
2º de DAM

Miguel Oliver
m.olivercano@edu.gva.es
2024/2025

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE

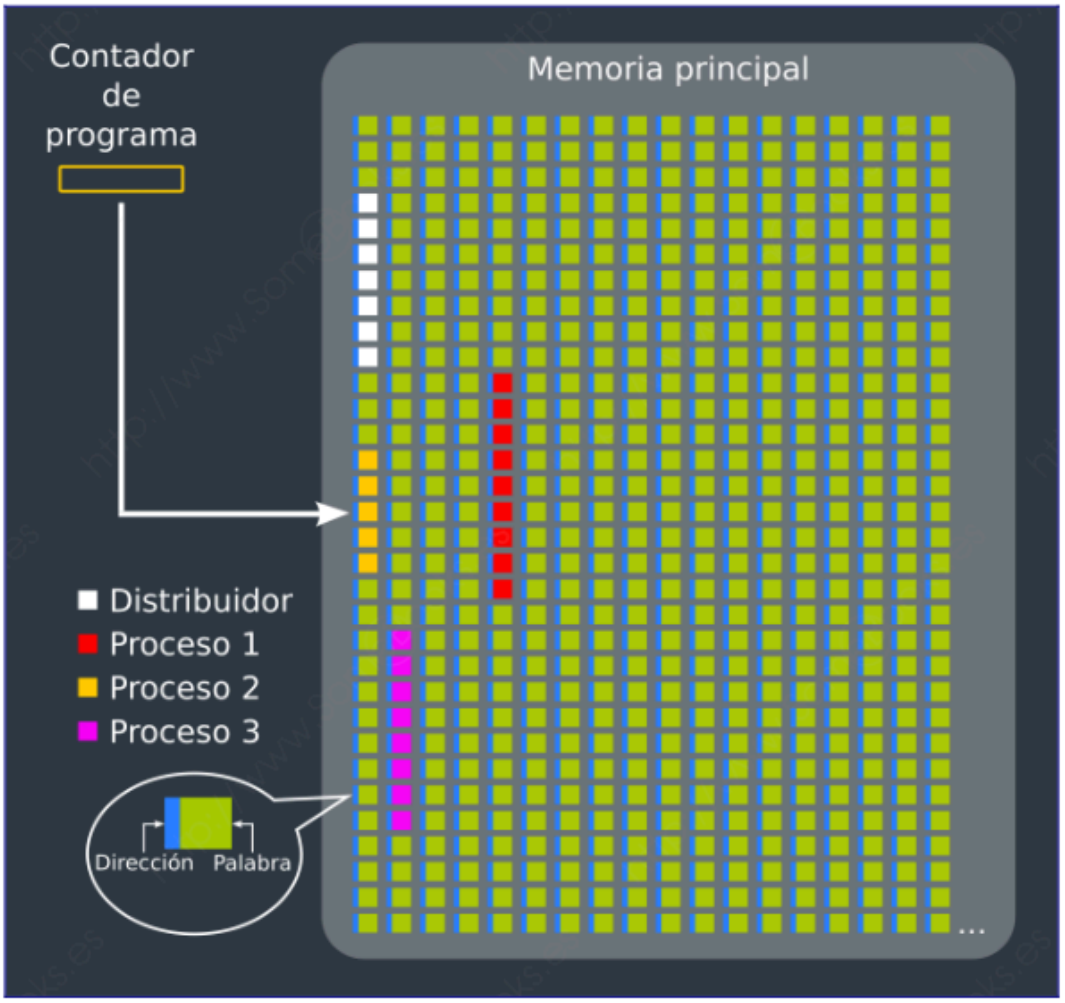
1. INTRODUCCIÓN

Breve descripción	ACTIVIDADES - PLANIFICACIÓN DE PROCESOS
Formato	PDF

2. ¿Cómo se ejecuta un proceso?

Cómo hemos dicho antes, porque un proceso se ejecute, su secuencia de instrucciones tiene que encontrarse en la memoria principal. Además, en todos los sistemas operativos modernos, se va intercalando la ejecución de diferentes procesos, de forma que se alternan el uso del procesador.
Para saber en qué posición de memoria se encuentra la siguiente instrucción que tiene que ejecutarse, el procesador dispone de un registro llamado Contador de programa (en inglés, programa Counter, o PC), que irá cambiando de valor según el paso del tiempo.

La secuencia de valores que vaya teniendo el Contador de programa podrán apuntar a instrucciones de diferentes procesos.



NOTA: El número de posiciones de memoria que ocupa un proceso es muy mayor del que representa la imagen. Se ha hecho así para simplificar la representación y facilitar la comprensión del gráfico.
NOTA: Un proceso puede crear otro proceso con el objetivo de estructurar su diseño o de aumentar su rendimiento. Por ejemplo, un servidor de impresión puede crear un nuevo proceso por cada documento a imprimir. De este modo, el proceso padre se encarga de la recepción de documentos y cada proceso hijo se centra en imprimir un documento concreto.

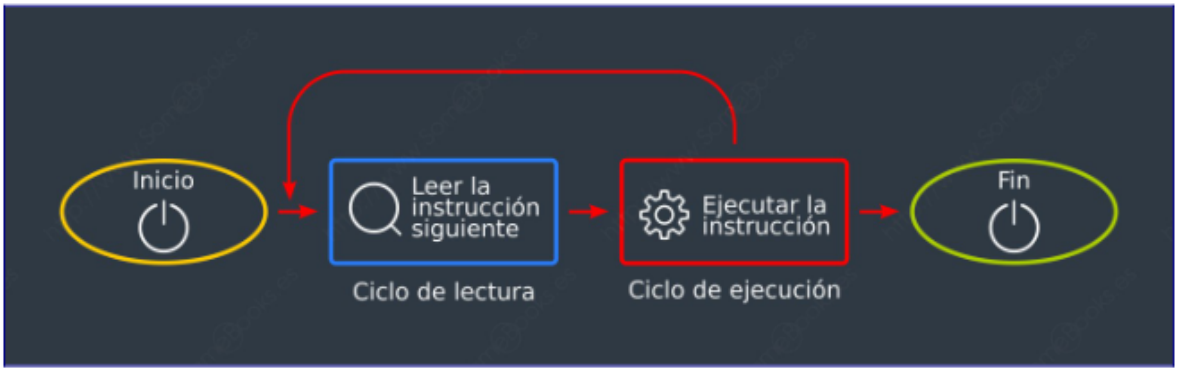
El procesador ejecutará el código perteneciente a un módulo del sistema operativo, llamado Distribuidor (en inglés, Dispatcher), cada vez que un proceso haya consumido su tiempo (medido en ciclos de instrucción) o haya solicitado algún servicio por el cual tenga que esperar (p. ej. una operación de E/S).

Cómo sabrás del módulo de hardware, podemos definir un ciclo de instrucción como el tiempo que emplea el procesador a ejecutar una instrucción en lenguaje máquina y, de una manera simplificada, podríamos dividirlo en dos pasos:

• El ciclo de lectura (en inglés, fetch), que consiste en cargar una instrucción desde la memoria principal a los registros del procesador.

• El ciclo de ejecución (en inglés, execute), que consiste en interpretar la instrucción (decodificarla) y ejecutarla, enviando las señales adecuadas a los componentes que tienen que realizar la operación que indica la instrucción.

Por este motivo, también suele decirse ciclo de fetch-and-execute o fetch-decode-execute.



RECUERDA: Decimos multitarea o multiprogramación a la capacidad que tienen los sistemas operativos actuales de alternar el uso del procesador entre diferentes procesos. Dada la velocidad a la cual funcionan los procesadores, el usuario tiene la sensación que los procesos se ejecutan al mismo tiempo.

Por otro lado, cuando en un sistema informático disponemos de varios procesadores (o incluso un único procesador con varios núcleos), pueden ejecutarse varios procesos al mismo tiempo. A esta técnica la decimos multiproceso o multiprocesamiento. Cuando todos los procesadores (o núcleos) actúan en igualdad de condiciones, hablamos de multiproceso simétrico o SMP (del inglés Symmetric Multe-Processing). Cuando el sistema dispone de procesadores con funciones especializadas, hablamos de multiproceso asimétrico o AMP (del inglés, Asymmetric Multe-Processing).

3. ¿Cómo se intercalan los procesos?

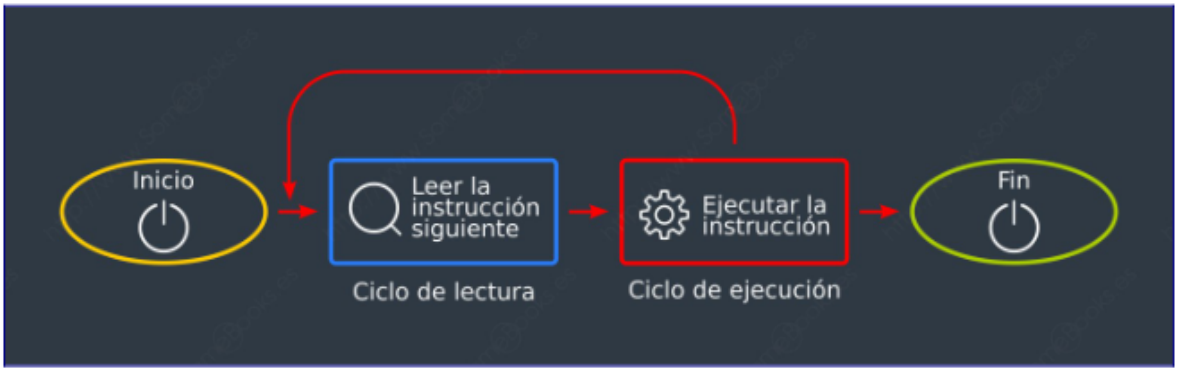
Ya hemos dicho más arriba que el procesador ejecutará el código perteneciente a un módulo del sistema operativo, llamado Distribuidor, cada vez que un proceso haya consumido su tiempo o haya solicitado algún servicio por el cual tenga que esperar. Así se evita que un proceso se apropie del procesador de manera indefinida.

De la idea anterior, podemos deducir que un proceso en particular puede encontrarse en tres situaciones diferentes:

• En ejecución: En este estado se encontrará el proceso que ocupa la atención del procesador en ese momento. Si el ordenador dispone de varios procesadores, o varios núcleos, podrá existir un proceso en ejecución por cada uno de los núcleos presentes.

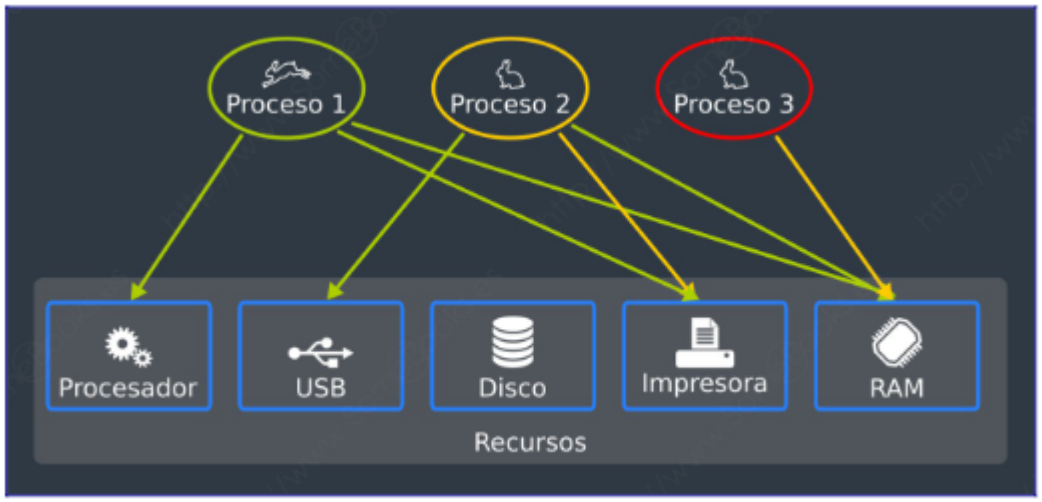
• Preparado: En este estado se encuentran los procesos que no se están ejecutando, pero que podrían hacerlo en cualquier momento y solo esperan su oportunidad para hacerlo.

• Bloqueado: En este estado estarán los procesos que han solicitado algún servicio del sistema operativo y están esperando una respuesta.

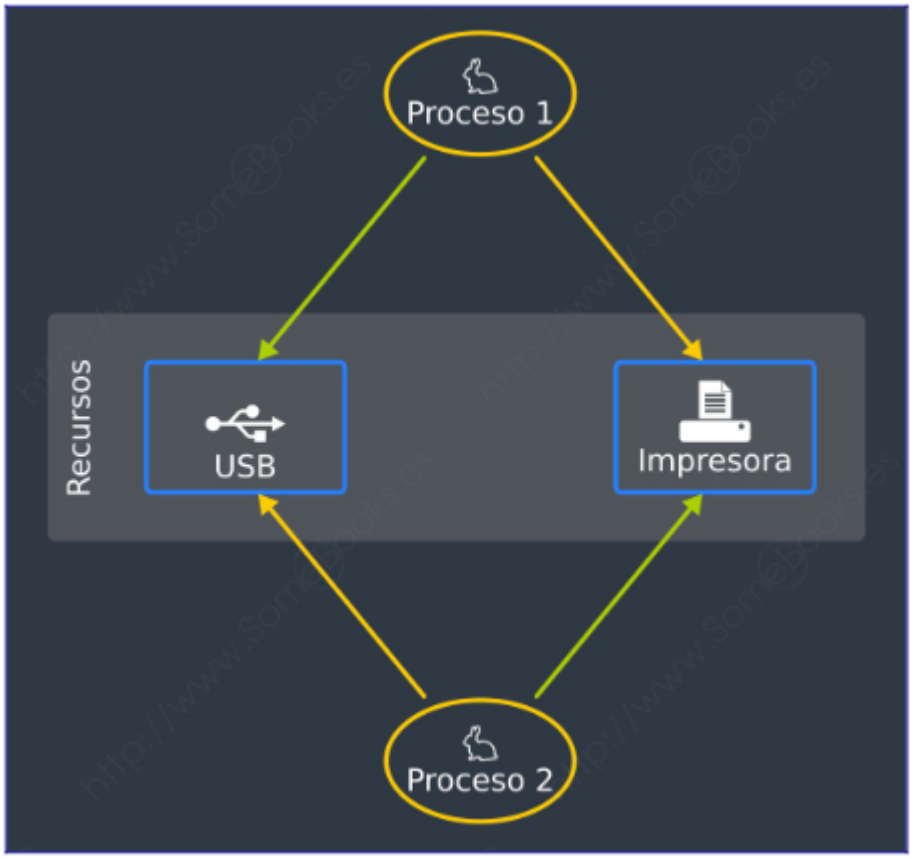


PARA SABER MÁS: Aunque no entraremos en muchos detalles, también podríamos definir un estado Nuevo en el cual se encontraría un proceso cuando acaba de crearse.
En esta situación, el proceso todavía no estaría incluido en el grupo de procesos Preparados y el sistema operativo todavía no se habría comprometido a ejecutarlo. Tampoco estaría cargado en la memoria.
Durante este estado, el sistema analizaría si dispone de los recursos necesarios para admitir su ejecución.
También podríamos tener procesos que se encuentran descargados a memoria secundaria, de manera temporal, con el objetivo de liberar la porción que ocupaban de memoria principal. En este caso, diremos que se encuentra en estado Suspenso (volveremos a hablar de este aspecto en el apartado destinado a la Gestión de memoria principal).

En la siguiente imagen vemos un proceso que se encuentra actualmente en ejecución, otro que se encuentra bloqueado en espera de poder utilizar la impresora y un tercero que se encuentra suspenso (no está usando la memoria principal, aunque la tiene solicitada):



En relación a esto, es importante que el sistema operativo trate de evitar el interbloqueo (en inglés, deadlock), que consiste en el hecho que dos procesos diferentes necesiten los dos mismos recursos y cada uno de ellos tiene uno asignado. Los dos estarán bloqueados en espera que se libere el recurso que todavía no tienen y, por lo tanto, ninguno liberará el recurso que ya poseen.

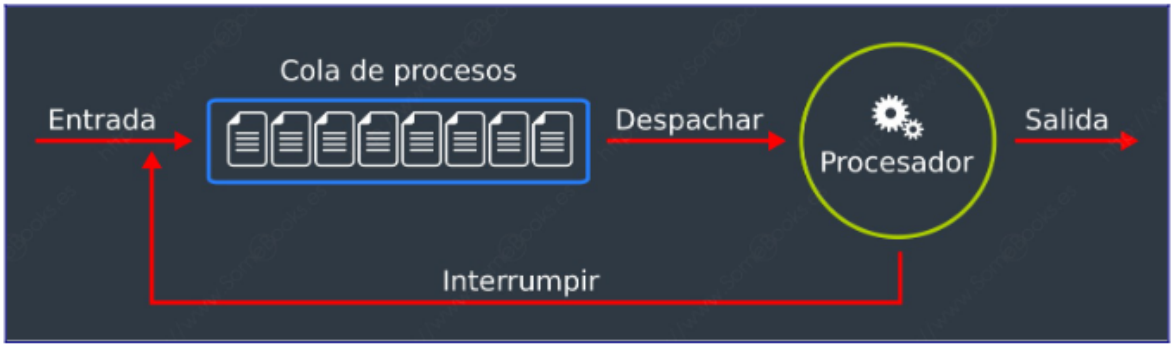


Aunque el ejemplo más evidente de interbloqueo se produce entre dos procesos, las situaciones más difíciles de predecir y solucionar implican múltiples procesos y recursos.

Cada vez que se crea un nuevo proceso, este es situado en estado de Preparado.

Cuando el proceso que se está ejecutando es interrumpido, el Distribuidor elige un nuevo proceso entre los cuales se encuentran en estado Preparado. El estado del proceso elegido pasa a ser En ejecución mientras que el proceso que abandona la ejecución pasará en estado Preparado (si ha consumido su tiempo) o Bloqueado (si ha realizado una petición al sistema).

Los procesos que se encuentran en estado Preparado esperan su turno en una cola.



PARA SABER MÁS: Decimos Traza de un proceso al listado de la secuencia de instrucciones que se ejecutan para este proceso. Se puede estudiar el comportamiento que está teniendo un procesador analizando la manera en que se intercalan las trazas de los diferentes procesos activos.

Cuando se ejecuta el módulo del kernel que se encarga de parar la ejecución de un proceso y hacer los cambios necesarios porque se ejecute un proceso diferente, decimos que se ha producido un cambio de contexto.

Un cambio de contexto lleva a cabo las siguientes acciones:
1. Guarda en la memoria principal el valor de los registros del procesador para el proceso que se estaba ejecutando.

2. Recupera el valor de los registros del procesador, desde la memoria principal, para el proceso que toma el relevo. El proceso elegido dependerá del Planificador del sistema operativo, que aplicará una determinada política para elegirlo (turno, prioridad, etc.).

3. Se ejecuta la instrucción indicada en el Contador de Programa, que forma parte del contexto que acabamos de recuperar y, por lo tanto, será la siguiente del nuevo proceso.

PARA SABER MÁS:

Cómo podemos deducir del que se ha dicho hasta ahora, los posibles cambios de estado son:

- Inicio a Nuez: Se crean las estructuras de datos del proceso que representa en un programa, para poder ejecutarlo.

- Nuevo a Preparado: Se produce cuando el sistema está listo para aceptar un nuevo trabajo. Los sistemas operativos suelen limitar la llegada de procesos nuevos para evitar que se degrade el rendimiento del sistema.

- Preparado a En ejecución: El sistema operativo elige uno de los procesos en estado Preparado.

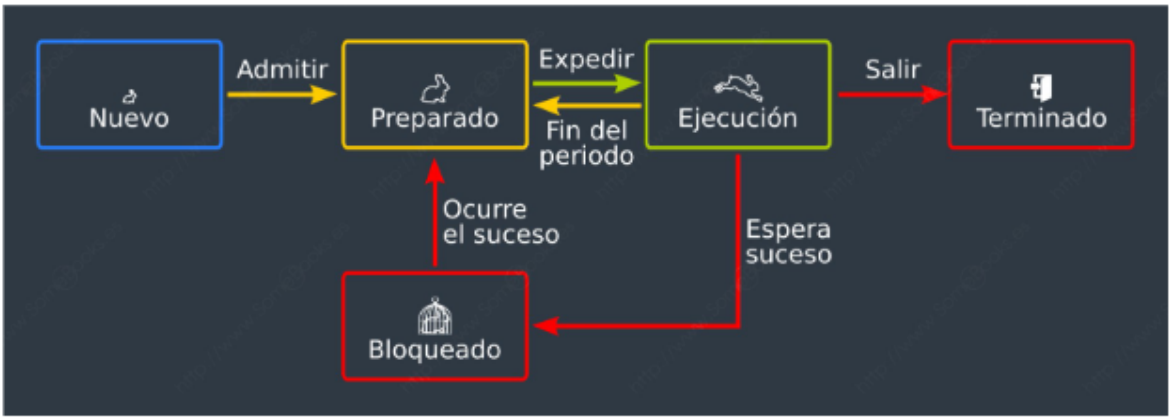
- En ejecución a Acabado: El proceso se finaliza cuando acaba su tarea o si lo abandona su proceso padre.

- En ejecución a Preparado: El proceso ha consumido el tiempo de ejecución que tenía asignado por el sistema operativo. También puede haber cedido el control o haber sido interrumpido por un proceso con una prioridad más elevada.

- En ejecución a Bloqueado: El proceso solicita un recurso (normalmente a través de una llamada a un servicio del sistema operativo) y no puede serle asignado en ese momento.

- Bloqueado a Preparado: El sistema operativo se encuentra en disposición de otorgar el recurso solicitado por el proceso.

- Preparado a Acabado/Bloqueado a Acabado: Cuando se trata de un proceso hijo, su proceso padre puede finalizarlo en cualquier momento. También puede finalizar un proceso hijo cuando finaliza su proceso padre.



4. ¿Cuándo acaba un proceso?

Todos los sistemas operativos tienen que tener un mecanismo para identificar cuando acaba un proceso. Si se trata de un script o un proceso por lotes (batch) concluirá cuando acaban sus instrucciones o cuando se encuentre una orden de parada (Halt). Si es un proceso interactivo, será el usuario el que elija el momento de acabar.

Además, un proceso puede verse interrumpido abruptamente por varios motivos. Entre ellos, podemos encontrar los siguientes:

• Sobrepasar el tiempo de ejecución asignado al proceso (tiempo real, de uso del procesador, etc.) o el tiempo máximo de espera ante un suceso.

• No disponer de memoria suficiente para satisfacer las solicitudes del proceso
• Que el proceso trate de acceder a posiciones de memoria o recursos del sistema que no tiene autorizados.
• Que una de sus instrucciones contenga un error aritmético o los datos no sean del tipo o tamaño adecuado.
• Que surja un error en una operación de entrada/salida (no existe un archivo, se produce un error de lectura, etc.)
• Que una instrucción del programa no exista en el juego de instrucciones o que sea una instrucción reservada al sistema operativo.
• Que el sistema operativo, el usuario o el proceso padre decide acabarlo. También suelen acabar los procesos hijos cuando acaba el proceso padre.

Lógicamente, cuando un proceso acaba, abandona su estado (En ejecución, Preparado, Bloqueado) y es eliminado de la cola o colas que dependen del Distribuidor.

5. Planificación de procesos

Una de las claves porque un sistema multiprogramado sea eficaz es la Planificación de procesos, que consiste a ir asignando procesos al procesador (o procesadores / núcleos) a lo largo del tiempo, de forma que se cumplan los objetivos en varios aspectos:

• Rendimiento: Trata de maximizar el número de acciones que se completan en un plazo de tiempo determinado.

• Tiempo de respuesta: El sistema tiene que responder a las solicitudes de los usuarios en un tiempo adecuado.

• Tiempo de retorno: El sistema tiene que ofrecer resultados de los procesos por lotes en un tiempo adecuado.

• Equidad: Todos los procesos tienen que ser considerados según sus características.

• Eficiencia: Se tiene que aspirar al hecho que el procesador esté activo constantemente. Como es debido esperar, el módulo del sistema operativo que se encarga de esta tarea se denomina Planificador (en inglés, Scheduler).

PARA SABER MÁS:
Cuando un proceso acaba, puede definirse un nuevo estado, del cual no hemos hablado todavía. Lo gritaremos Acabado y será el estado en el cual se encuentre un proceso que ha sido eliminado del grupo de procesos ejecutables por cualquiera de los motivos anteriores. Este estado se mantendrá mientras se liberan sus tablas de información, su memoria y cualquier otro dato que haya sido necesario para su ejecución.

Si el proceso colabora con otros, también puede almacenar información necesaria para ellos, y se les dará la oportunidad de obtenerlos antes de que sean eliminados.

Según el diseño del sistema operativo, el Planificador utilizará unos criterios u otros para llevar a cabo su tarea. Estos criterios reciben el nombre de Algoritmos de Planificación (o también, Políticas de Planificación).

A continuación, nombramos los más importantes:

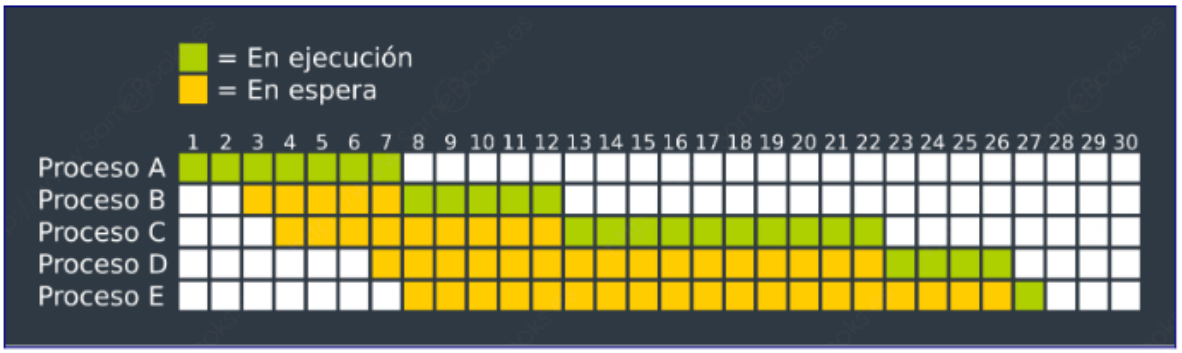
6. Primero a llegar primero a ser servido, o FCFS (del inglés, First Come First Served)

Se emplea en procesos por lotes (sin intervención del usuario) y es no apropiativo. Los procesos se van poniendo en cola según llegan y se les asigna el estado Preparado. Cuando es asignado al procesador, no lo abandona hasta que acaba.
Una de sus ventajas principales consiste en el hecho que es un algoritmo muy sencillo de implementar y también es fácilmente predecible.
Entre sus principales inconvenientes podemos mencionar que los procesos largos pueden hacer esperar mucho a los procesos cortos y que el tiempo de servicio mínimo variará muy según el número de procesos ejecutados y la duración de estos. Además, el tiempo medio de espera depende del hecho de que llegan antes los procesos más cortos o los más largos.

Para entender su funcionamiento, podemos estudiar el siguiente ejemplo práctico en el cual disponemos de cinco procesos que se incorporan al sistema en momentos sucesivos y que tienen diferentes tiempos de duración:

Planificación FCFS		
Proceso	**Momento de llegada**	**Duración**
A	1	7
B	3	5
C	4	10
D	7	4
E	8	1

Si analizamos de manera gráfica el comportamiento del sistema, observaremos que se comporta de la siguiente manera.



En consecuencia, podemos extraer las siguientes conclusiones:

Planificación FCFS						
Proceso	Momento de llegada	Duración	Momento de comienzo	Momento de terminación	Tiempo de respuesta	Tiempo de espera
A	1	7	1	7	7-0 = 7	0
B	3	5	8	12	12-2 = 10	10-5 = 5
C	4	10	13	22	22-3 = 19	19-10 = 9
D	7	4	23	26	26-6 = 20	20-4 = 16
E	8	1	27	33	33-7 = 26	26-1 = 25

Actividad 1: Planificación FCFS

Supone que tenemos tres usuarios y que cada uno necesita ejecutar un proceso por lotes diferente:

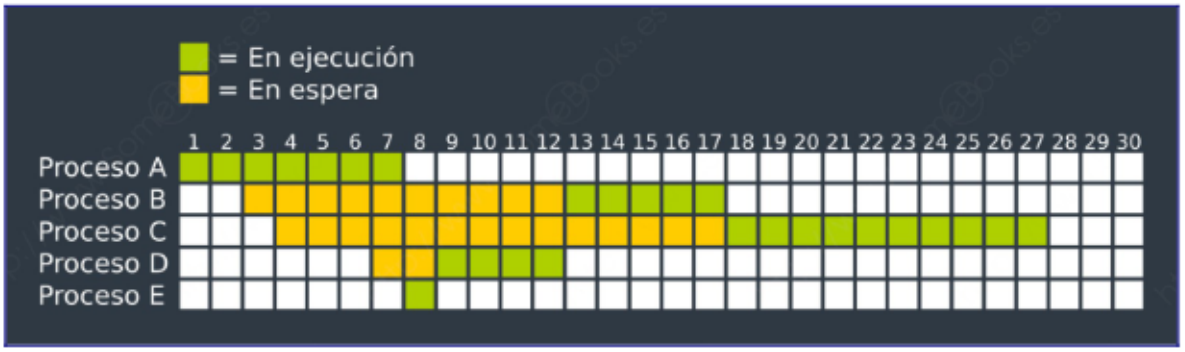
- El primero necesitará 18 minutos para ejecutarse.
- El segundo necesitará 6 minutos.
- El tercero duró otros 7 minutos.

Sabiendo que el tiempo medio de respuesta será la suma de los tiempos que espera cada usuario para obtener sus resultados, dividido entre tres, tendrá alguna importancia del orden en el cual se planifican los procesos si utilizamos un algoritmo FCFS y solo disponemos de un procesador?

Para ilustrar tu conclusión, tendrás que crear gráficas con las diferentes combinaciones y calcular el tiempo medio de respuesta en cada caso.

7. Primero el más corto, o SJF (del inglés, Shortest Job First)

Se emplea en procesos por lotes (sin intervención del usuario) y es no apropiativo. Los procesos se van poniendo en cola según llegan y se los asigna el estado Preparado, pero el Planificador elige el que tiene un menor tiempo previsto de ejecución.
Cómo en el caso anterior, podemos estudiar un ejemplo práctico para entender su funcionamiento. En este caso, partiremos de los mismos datos, para que puedas comparar los resultados:



Y analizando los resultados, obtendremos la siguiente tabla:

Planificación SJF						
Proceso	**Momento de llegada**	**Duración**	**Momento de comienzo**	**Momento de terminación**	**Tiempo de respuesta**	**Tiempo de espera**
A	1	7	1	7	7-0 = 7	0
B	3	5	13	17	17-2 = 15	15-5 = 10
C	4	10	18	27	27-3 = 24	24-10 = 14
D	7	4	9	12	12-6 = 6	6-4 = 2
E	8	1	8	8	8-7 = 1	1-1 = 0

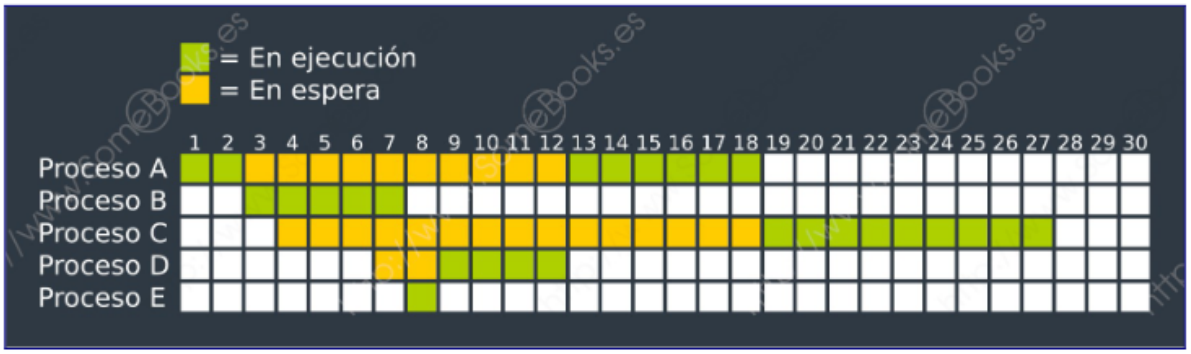
Existe una versión apropiativa de este algoritmo denominada Primero el de menor tiempo restante, o SRTN (del inglés, Shortest Remaining Time Next). En este caso, si un proceso bloqueado pasa en el estado Preparado, el distribuidor comprueba si su tiempo restante es inferior que el del proceso que se encuentra en ejecución. En caso afirmativo, este toma el control del procesador y el proceso que está ejecutándose pasa al estado Preparado.

Aunque es un algoritmo muy eficaz para los procesos cortos, resulta difícil predecir los intervalos de asignación del procesador e, incluso, puede haber procesos largos que sufren de inanición, es decir, que no llegan a ejecutarse mientras existan procesos cortos esperando turno.

Como antes, lo ilustraremos con un ejemplo. En este caso, hemos cambiado ligeramente los datos iniciales para que sea más ilustrativo:

Planificación SRTN		
Proceso	**Momento de llegada**	**Duración**
A	1	8
B	3	5
C	4	9
D	7	4
E	8	1

Y el gráfico resultante quedará como sigue:



De nuevo, aquí tenso los datos lanzados, representados en forma de mesa:

Planificación SRTN						
Proceso	**Momento de llegada**	**Duración**	**Momento de comienzo**	**Momento de terminación**	**Tiempo de respuesta**	**Tiempo de espera**
A	1	8	1	18	18-0 = 18	18-8 = 10
B	3	5	3	7	7-2 = 5	5-5 = 0
C	4	9	19	27	27-3 = 24	24-9 = 15
D	7	4	9	12	12-6 = 6	6-4 = 2
E	8	1	8	8	8-7 = 1	1-1 = 0

Actividad 2: Planificación SRTN

Supone que tenemos tres usuarios y que cada uno necesita ejecutar un proceso por lotes diferente:

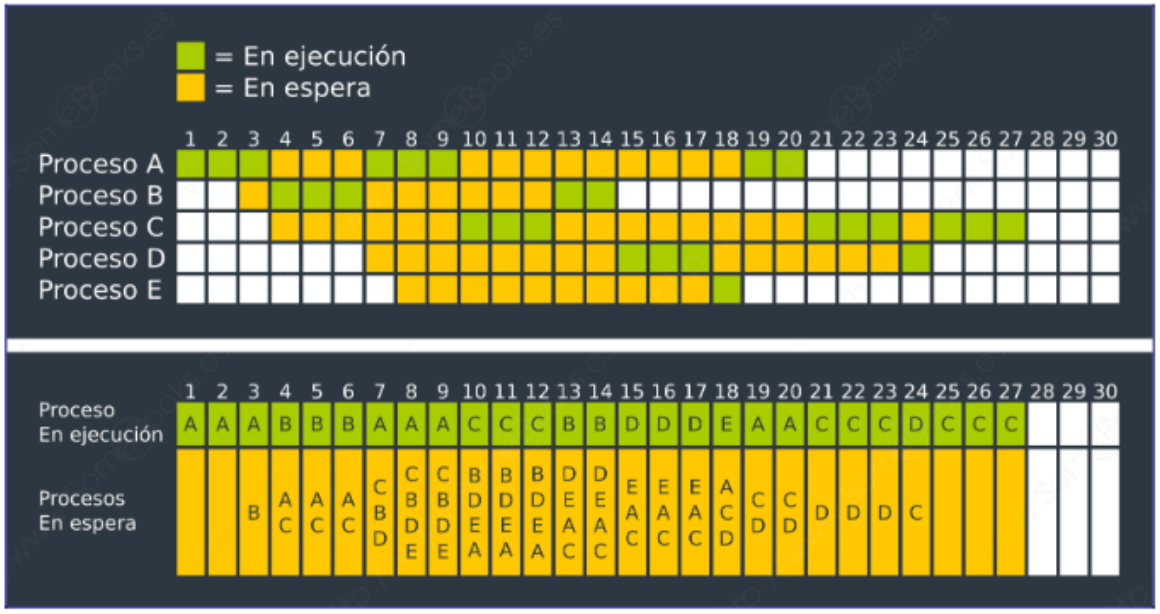
- El primero necesitará 18 minutos para ejecutarse, y es el único que está listo al principio.
- El segundo necesitará 10 minutos y estará disponible a partir del minuto 7.
- El tercero necesitará 6 minutos y estará disponible a partir del minuto 21.

A partir de estos datos, crea una gráfica donde se aprecie la orden de ejecución de los procesos suponiendo que usamos el algoritmo SRTN si el Planificador toma el control del sistema una vez por minuto (y desestimamos el tiempo que se está ejecutando).

A continuación, crea una mesa donde expreses el tiempo de respuesta y el tiempo de espera en función de los datos de entrada.

8. Por turnos, o RR (del inglés, Round Robin)

Se emplea en procesos interactivos (en los cuales interviene el usuario) y es apropiativo. Los procesos se van poniendo en cola según llegan y se les asigna el estado Preparado. El procesador se irá asignando a cada proceso, por orden, durante una fracción de tiempo denominada Quantum, que es igual para todos. Si el proceso se acaba, se bloquea, o si se agota su tiempo, el procesador es liberado para el siguiente proceso de la lista.
Por lo tanto, la cola de procesos actúa como una estructura circular con organización FIFO.
Para ilustrarlo, empezaremos por los mismos datos de entrada del ejemplo anterior, suponiendo un Quantum equivalente a tres unidades de tiempos. El resultado seria como el que muestra la siguiente imagen:



Para facilitar la comprensión, hemos incluido también una representación gráfica de los procesos que se encuentran en cola para ser ejecutados y del proceso que se encuentra en ejecución en cada unidad de tiempo.

Planificación Round Robin						
Proceso	**Momento de llegada**	**Duración**	**Momento de comienzo**	**Momento de terminación**	**Tiempo de respuesta**	**Tiempo de espera**
A	1	8	1, 7, 19	20	20-0=20	20-8=12
B	3	5	4, 13	14	14-2=12	12-5=7
C	4	9	10, 21, 25	27	27-3=24	24-9=15
D	7	4	15, 24	24	24-6=18	18-4=14
E	8	1	18	18	18-7=11	11-1=10

Finalmente, la siguiente tabla muestra los datos que ofrece:

Actividad 3: Planificación por turnos

Partiendo de los mismos datos de entrada que en la actividad anterior, aplica la planificación por turnos suponiendo un Quantum equivalente a tres unidades de tiempos. Tienes que hacer las siguientes tareas:

1. Crea un esquema, como el aportado en este documento, pero teniendo en cuenta los 'Momentos de llegada' y la 'Duración' de los procesos de la actividad anterior.

2. A continuación, crea una nueva versión de la tabla de planificación, ajustando su contenido al diseño del esquema creado en el apartado anterior.

9. Planificación por prioridad

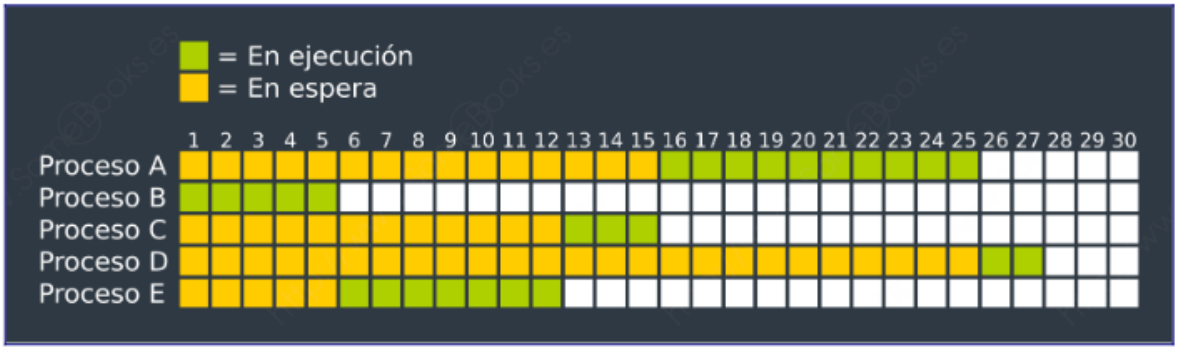
Se emplea en procesos interactivos (en los cuales interviene el usuario) y es apropiativo. A cada proceso se le asigna un número entero que representa su prioridad, de forma que, cuánto menor es el número, mayor es la prioridad.

Si se considera un algoritmo no apropiativo, funciona como el algoritmo Primero lo más corto (SJF), pero considerando la prioridad en lugar de la duración.

Todavía así, veamos un ejemplo, suponiendo que todos los procesos están disponibles desde el primer momento. Estos serían nuestros datos de partida:

Planificación por prioridad		
Proceso	**Duración**	**Prioridad**
A	10	4
B	5	1
C	3	3
D	2	5
E	7	2

Siguiendo la asignación del procesador según la orden de prioridad (lo más bajo primero), obtendremos el siguiente gráfico:



Finalmente, en la siguiente tabla recogemos los datos numéricos obtenidos:

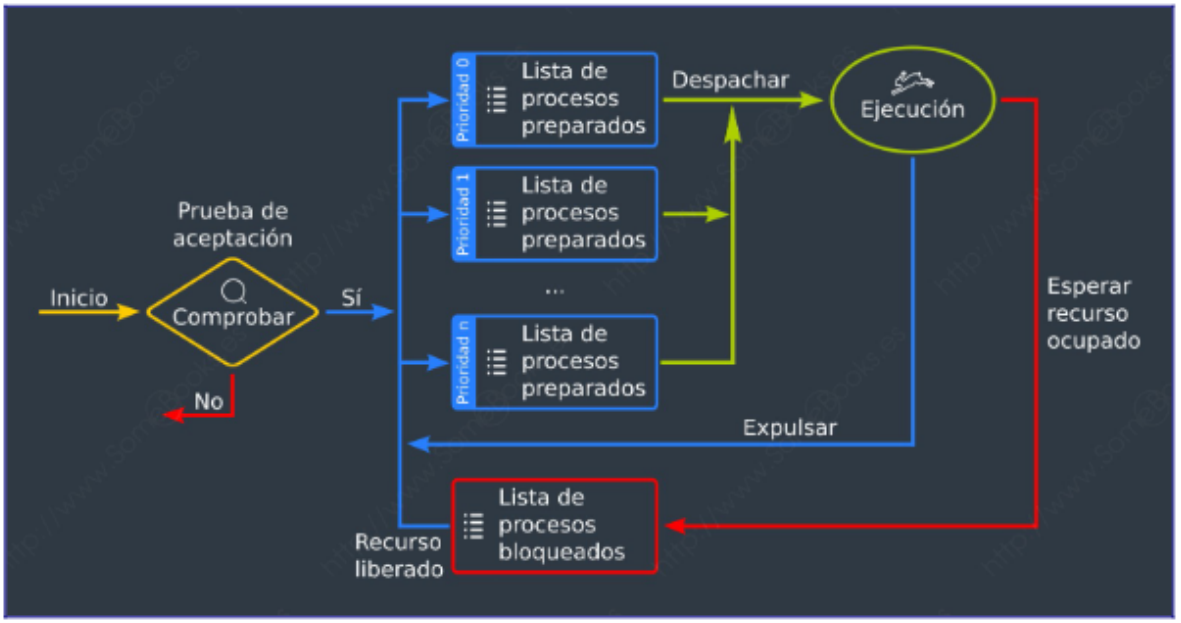
Planificación por prioridad						
Proceso	Duración	Prioridad	Momento de comienzo	Momento de terminación	Tiempo de respuesta	Tiempo de espera
A	10	4	16	25	25	25-10 = 15
B	5	1	1	5	5	5-5 = 0
C	3	3	13	15	15	15-3 = 12
D	2	5	26	27	27	27-2 = 25
E	7	2	6	12	12	12-7 = 5

Para evitar que los procesos con una prioridad más elevada monopolizan el uso del procesador (y los de prioridad más baja sufren inanición), la prioridad de los procesos puede ir reduciéndose cada vez que obtengan el uso del procesador. Así, se convertiría en una versión apropiativa que funciona como Primero el de menor tiempo restando (SRTN), pero tomando la prioridad en lugar de la duración y disminuyendo cada vez.

PARA SABER MÁS: Cómo hemos comentado más arriba, en el supuesto de que los procesos puedan tener diferentes prioridades, el Planificador elegirá siempre al proceso de mayor prioridad para su ejecución. Y en el supuesto de que existan varios procesos con el mismo grado de prioridad, es común que se establezca un turno rotatorio entre ellos.

La manera de implementar esta característica puede consistir en mantener una cola de procesos en estado Preparado para cada uno de los posibles niveles de prioridad. Cómo hemos dicho más arriba, para evitar que los procesos con menor prioridad puedan sufrir inanición, en los procesos de mayor prioridad, esta puede ir degradándose en función de su antigüedad o de la frecuencia con la cual hayan obtenido el uso del procesador.

Por lo tanto, el esquema sobre los estados de un proceso que vimos al principio podría quedar como sigue:



10. Planificación por reparto equitativo, o FSS (del inglés, Fair-share Scheduling)

Se emplea en procesos interactivos (en los cuales interviene el usuario) y es apropiativo. Se tiene en cuenta el número de usuarios que serán atendidos, de forma que el tiempo de ejecución se reparte entre ellos de manera equitativa. La ejecución del conjunto de procesos de cada usuario no sobrepasará el tiempo asignado a este usuario.

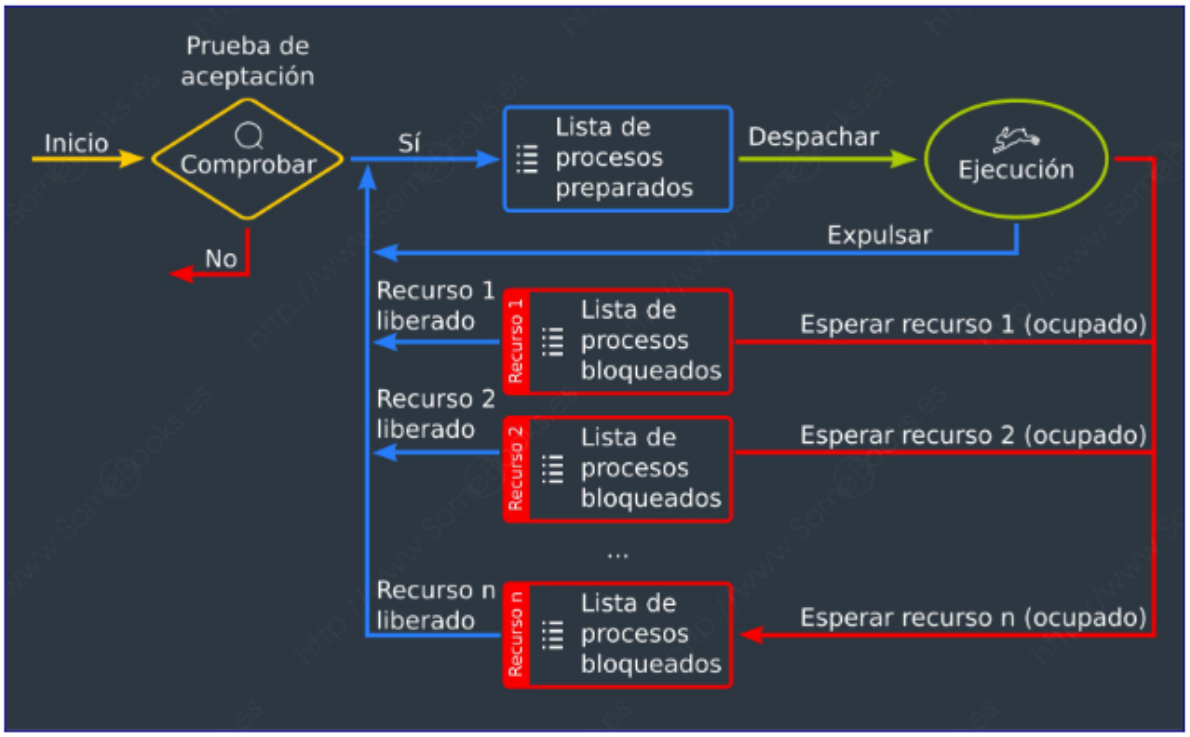
Este concepto puede ampliarse a grupos de usuarios, de forma que las decisiones de planificación otorgan a cada grupo de usuarios un servicio equivalente. De este modo, el uso intensivo del sistema por parte de un grupo de usuarios no perjudica a los otros grupos.

11. Planificación de Colas Múltiples, o MQS (del inglés, Multilevel Queue Scheduling)

Se emplea tanto en procesos interactivos como en procesos por lotes (en los cuales no interviene el usuario) y es apropiativo. Consiste en fragmentar la cola de procesos en estado Preparado en varias colas más pequeñas, de forma que cada una puede estar administrada por

un algoritmo de planificación diferente. De este modo, cada proceso será asignado a una determinada cola en función de sus características pudiendo tratar de manera diferente, por ejemplo, a los procesos interactivos y a los procesos por lotes.

PARA SABER MÁS: Del mismo modo que en algunas situaciones puede ser interesante contemplar diferentes colas de procesos en estado Preparado, puede ser conveniente disponer de una cola de procesos en estado Bloqueado para cada tipo de suceso. De este modo, el sistema evita recorrer toda la lista de bloqueados para localizar el proceso que está esperando un suceso concreto.



12. Comunicación entre procesos

En muchas ocasiones, un sistema operativo ejecuta varios procesos que tienen que comunicarse entre ellos para colaborar en un objetivo común. Para conseguirlo, el sistema puede ofrecer una serie de funciones llamadas IPC (del inglés, Inter-Process Communication) que facilitan el envío de mensajes entre los procesos para comunicarse y sincronizarse.
Otra forma que tienen los procesos de comunicarse entre sí es compartiendo determinadas zonas de memoria.

Cuando los procesos que necesitan comunicarse están en ordenadores diferentes, se utiliza RPC (del inglés, Remote Procedure Call). Sin embargo, quien programa estos procesos no

tendrá que preocuparse de su ubicación física y de los aspectos que derivan de esa circunstancia. Para ellos, no habrá diferencia con la comunicación con procedimientos locales. Es decir, será el sistema operativo quién se encargue de resolver los problemas que se derivan de esta situación.