SISTEMAS OPERATIVOS

3004610 - 1

German Sánchez Torres, I.S., M.Sc., Ph.D.

Profesor, Facultad de Ingeniería - Programa de Sistemas

Universidad del Magdalena, Santa Marta.

Phone: +57 (5) 4214079 Ext 1138 - 301-683 6593

Edificio Docente, Cub 3D102.

Email: sanchez.gt@gmail.com –gsanchez@unimagdalena.edu.co

Procesos

SISTEMAS OPERATIVOS

3004610 - 1

PROCESOS

1. Concepto de proceso
2. Multitarea
3. Información de los procesos
4. Estado de los procesos

1. Concepto de proceso

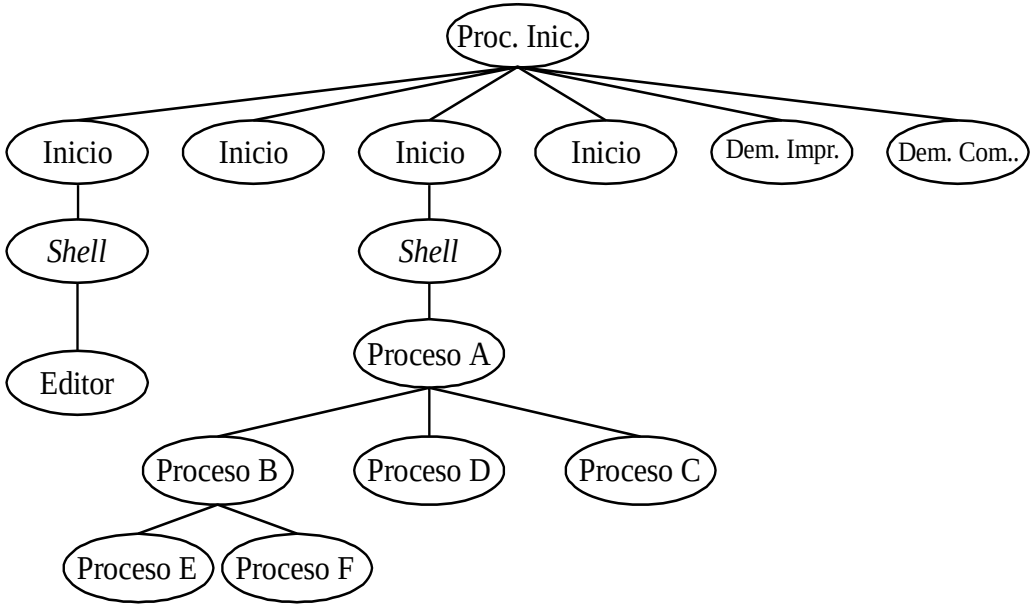
Proceso:

- Programa en ejecución. Unidad de procesamiento gestionada por el SO
- El SO mantiene para cada proceso una serie de estructuras de información que permiten identificar sus características y los recursos asignados. Una parte básica de esta información se encuentra en el BCP (***bloque de control de proceso***). El SO mantiene una tabla con los BCPs de todos los procesos

Información asociada a cada proceso:

- Contenido de los segmentos de memoria donde residen el código y los datos (*core image* o imagen de memoria)
- Contenido de los registros del modelo de programación (PC, estado, etc.)
- Contenido del BCP

Ya que los procesos encargados de arrancar el SO son los primeros en ejecutar, se va creando una jerarquía de procesos como la indicada a continuación



Para referirse a las relaciones entre procesos se emplean los términos **padre, hijo, hermano, etc**. Cuando un proceso A solicita al SO la creación de un nuevo proceso (B), se dice que A es padre de B y que B es hijo de A. UNIX mantiene explícitamente una estructura jerárquica de procesos.

El **entorno del proceso** consiste en un conjunto de variables que se le pasan al nuevo proceso en el momento de su creación. Está formado por una tabla de pares NOMBRE-VALOR (nombre de variable, valor de variable). Algunos ejemplos de variables de entorno son:

- PATH=/usr/local/bin:/usr/bin
- TERM=xterm
- HOME=/home/fc2
- PWD=/home/fc2/documentos

Los **procesos forman grupos**, con diferentes propiedades. El conjunto de procesos creados a partir de un *shell* puede forman un grupo. También puede ser un grupo el conjunto de procesos asociados a un usuario o a un terminal. Algunos servicios del SO pueden operar selectivamente sobre todos los procesos de un grupo.

2. Multitarea

2.1. Introducción
2.2. Base de la multitarea
2.3. Ventajas
2.4. Grado de multiprogramación y necesidades de MP

Introducción

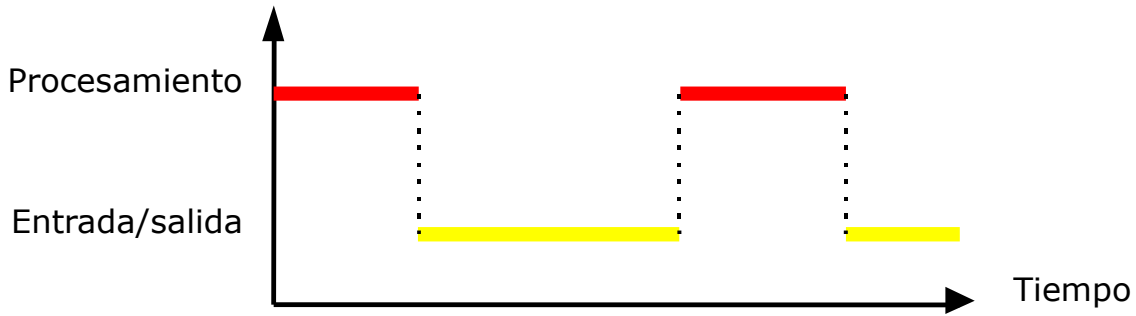
Dependiendo del número de procesos y usuarios que pueden ejecutar simultáneamente, un SO puede ser:

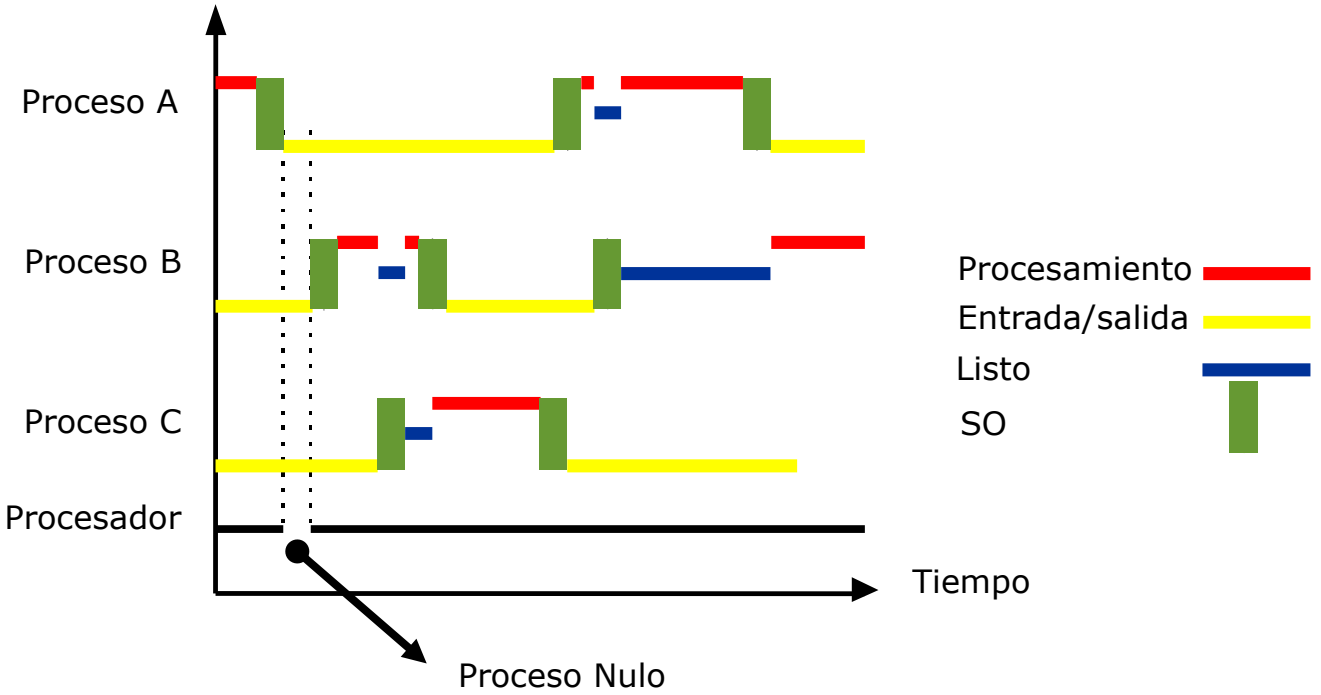
- **Monotarea** (monoproceso): sólo un proceso en cada instante
- **Multitarea** (multiproceso)
- **Monousuario**
- **Multiusuario** (tiempo compartido)

Nº usuarios \ Nº procesos	1	más de 1
1	Monoproceso Monousuario	Multiproceso Monousuario
más de 1	———	Multiproceso Multiusuario

La multitarea se basa en las siguientes características:

a) **Memoria principal** capaz de almacenar **varios procesos**
b) **Paralelismo real entre operaciones de E/S y uso del procesador**: cuando un proceso realiza una operación de E/S se pasa a ejecutar otro
c) **Alternancia en los procesos en fases de E/S y procesamiento**: aprovechar las fases de realización de operaciones de E/S de unos procesos para ejecutar otros




Proceso A
Proceso B
Proceso C
Procesador
Tiempo
Proceso Nulo
Procesamiento
Entrada/salida
Listo
SO

Ventajas de la multitarea

Las más importantes son:

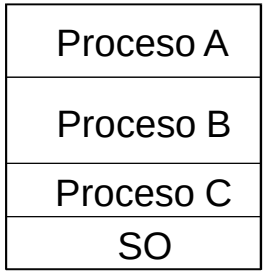
a) **Facilita la programación**. Las aplicaciones pueden organizarse en varios procesos, beneficiando así la modularidad.
b) Permite atender a **múltiples usuarios** de forma eficiente, interactiva y simultánea.
c) Aprovecha los **tiempos muertos** que los procesos pasan esperando la conclusión de operaciones de E/S.
d) Aumentar el uso de la CPU, al aprovechar los tiempos de bloqueo de unos procesos en la ejecución de otros.

Grado de multiprogramación y necesidades de MP

Grado de multiprogramación: número de procesos activos mantenidos por el SO. A más procesos activos, mayor probabilidad de encontrar en cada instante uno en situación de ser ejecutado. A más procesos activos mayores necesidades de memoria.

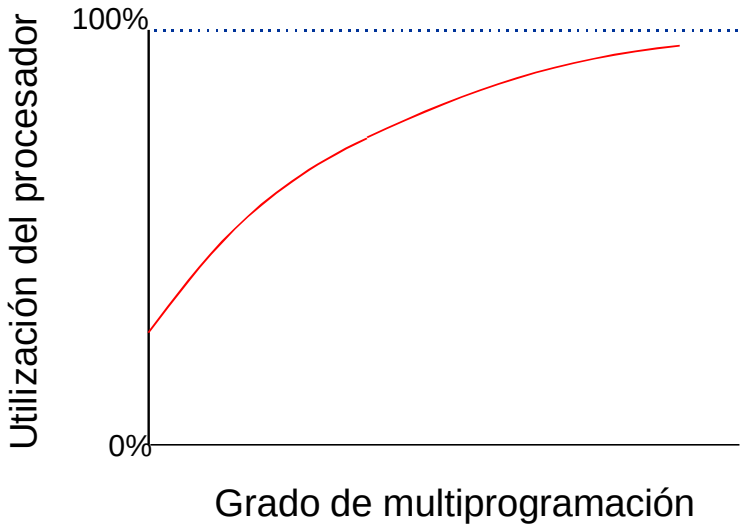
En un sistema **sin memoria virtual** los procesos activos han de residir plenamente en MP. En este caso, el grado de multiprogramación está severamente limitado por el tamaño de los procesos y por la memoria disponible. En este caso, a más procesos mayor grado de uso de la CPU (procesos siempre en MP).

Grado de multiprogramación y necesidades de MP (sin MV)



Memoria principal

Cada proceso reside totalmente en M.p

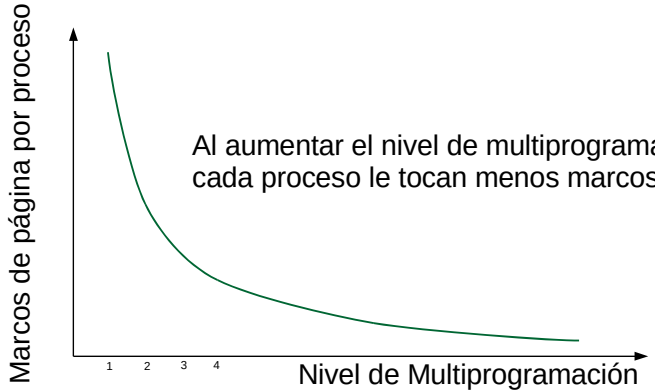


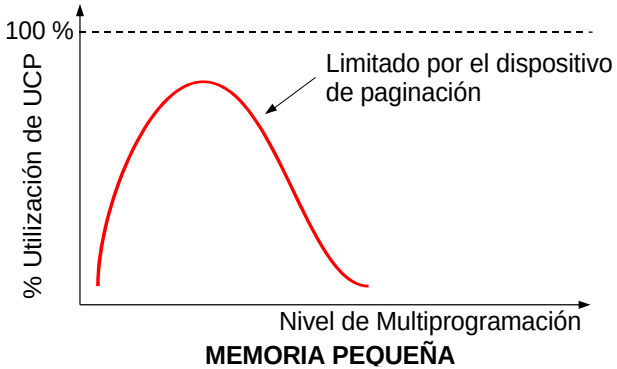
En sistemas **con memoria virtual** la situación es más compleja. Los procesos sólo tienen en MP su conjunto residente (páginas que están en MP), por lo que podríamos tener más procesos cargados en MP. Si se aumenta demasiado el número de procesos, el conjunto residente de cada uno se hace muy pequeño.

Al disminuir el conjunto residente ya no representa adecuadamente al futuro conjunto de trabajo (páginas en uso), por lo que se producirán muchos fallos de página.

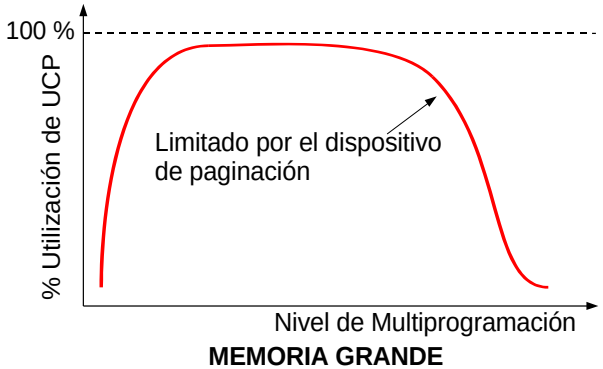
Los fallos de página consumen mucho tiempo del procesador, ya que el SO ha de tratar el fallo, y tiempo de E/S, ya que es necesaria una migración de páginas.

Al crecer el número de fallos de página sistema dedica más tiempo a resolver los fallos de página que a realizar las tareas de los programas (**hiperpaginación**).





MEMORIA PEQUEÑA



MEMORIA GRANDE

Información de los procesos

3.1. Estado del procesador
3.2. Imagen de memoria de un proceso
3.3. Información del BCP
3.4. Tablas del SO

Estado

El estado del procesador queda definido por **el contenido de sus registros**: registros generales, contador del programa, puntero de pila, registros de estado, registros especiales (RIED: ***registro identificador de espacio de direcciones***)

Cuando un proceso se ejecuta el estado del procesador responde a su ejecución, y se almacena directamente en los registros del procesador. Cuando el proceso deja de ejecutarse esta información pasa a almacenarse en el bloque de control de proceso (BCP).

La rutina del SO que trata las interrupciones, en primer lugar, salva el estado del procesador en el BCP del proceso interrumpido.

Imagen de memoria de un proceso

La imagen de memoria de un proceso está constituida por **los espacios de memoria que está autorizado a usar** (los que el SO le ha asignado). Algunas cuestiones a tener en cuenta son:

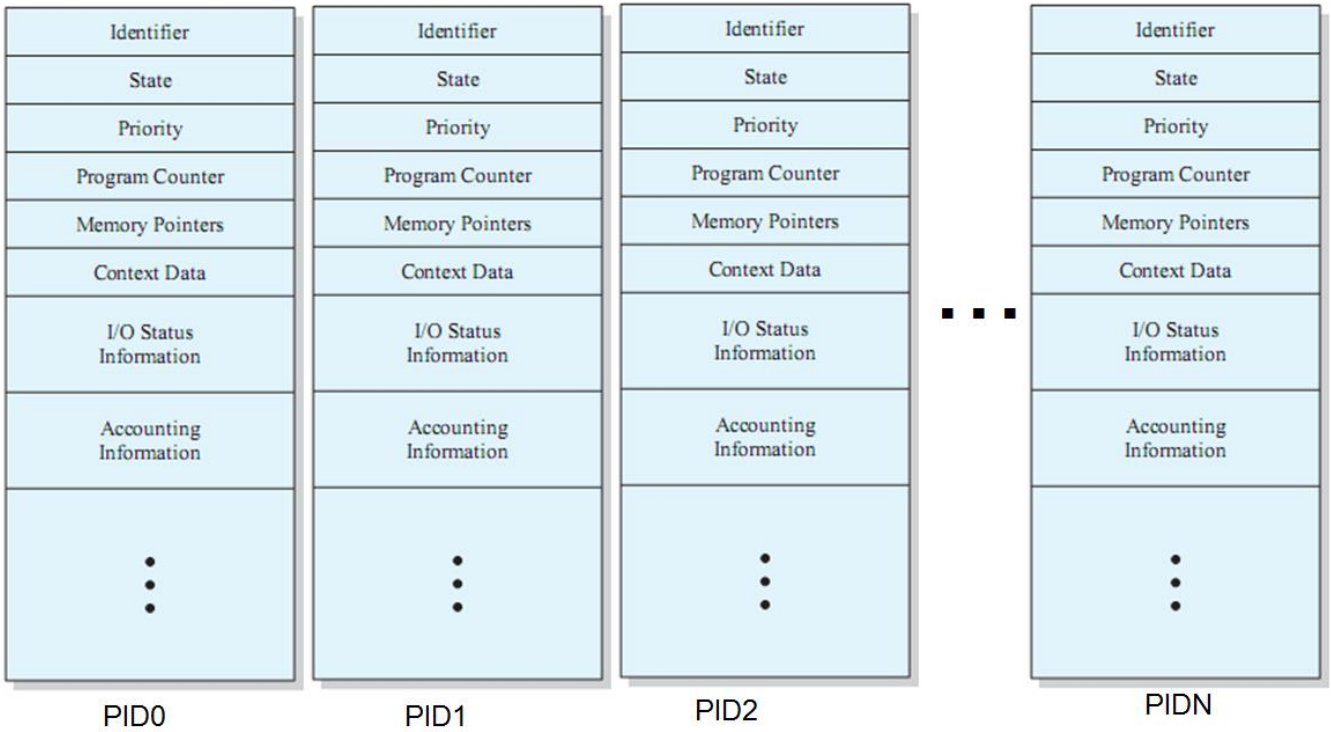
a) El proceso sólo puede tener información en su imagen de memoria. Si intenta un acceso a una posición de memoria fuera de ella, el hardware de **protección** lo detectará y generará la excepción correspondiente. Esta excepción activa la ejecución del SO, quien tomará la opción oportuna (generalmente abortar el proceso)

b) Los procesos suelen necesitar **asignación dinámica** de espacio. Por tanto, la imagen debe adaptarse a sus necesidades, creciendo o decreciendo

Información del BCP

- **Identifier:** A unique identifier associated with this process, to distinguish it from all other processes.
- **State:** If the process is currently executing, it is in the running state.
- **Priority:** Priority level relative to other processes.
- Program counter: The address of the next instruction in the program to be executed.
- **Memory pointers:** Includes pointers to the program code and data associated with this process, plus any memory blocks shared with other processes.
- **Context data:** These are data that are present in registers in the processor while the process is executing.
- **I/O status information:** Includes outstanding I/O requests, I/O devices (e.g., tape drives) assigned to this process, a list of files in use by the process, and so on.
- **Accounting information:** May include the amount of processor time and clock time used, time limits, account numbers, and so on.

Ejemplo simple de una tabla de BCPs



formación de los procesos

La **formación de un proceso** es el proceso por el que se completan todas las **informaciones que lo describen**. El SO realiza:

a) Asignación de espacio de memoria (normalmente espacio virtual en varios segmentos).
b) Seleccionar BCP libre en la tabla de procesos
c) Rellenar BCP con información de identificación del proceso, descripción de memoria asignada, valores de los registros, etc
d) Carga del segmento de texto, con el código.
e) Carga del segmento de datos iniciales del fichero objeto.
e) Inicialización de la pila en el segmento de pila. Inicialmente incluye el entorno del proceso y los parámetros pasados en la invocación del programa correspondiente.

Una vez hecho esto, el proceso puede marcarse como **listo**, de forma que el planificador, cuando lo considere oportuno, pase a ejecutarlo.

4. Estado de los procesos

4.1. Introducción
4.2. Cambio de contexto

Introducción

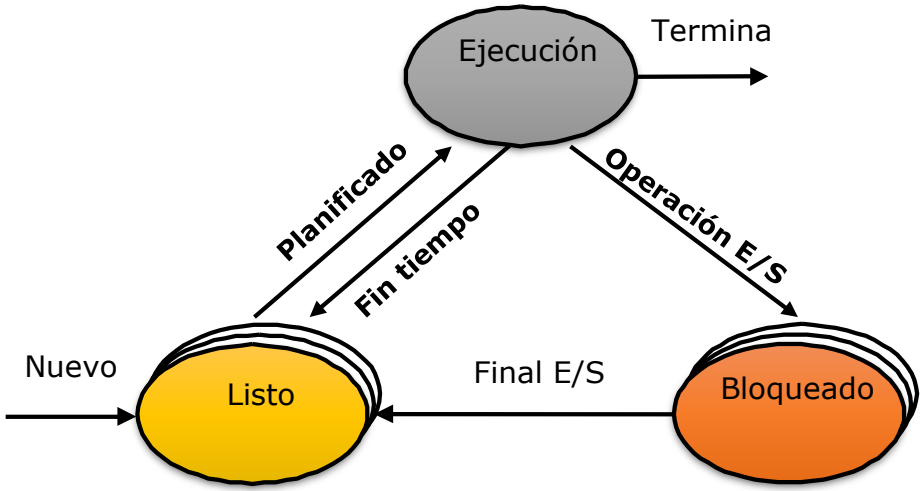
No todos los procesos activos de un sistema multitarea están en la misma situación. Se distinguen tres estados básicos:

a) Ejecución. En este estado sólo está el proceso que está siendo ejecutado. El estado del proceso reside en los registros del procesador.

b) Bloqueado. A la espera de que ocurra un evento. Situación típica: un proceso solicita una operación de E/S. Hasta que no finalice la operación el proceso queda bloqueado. El estado del proceso reside en su BCP

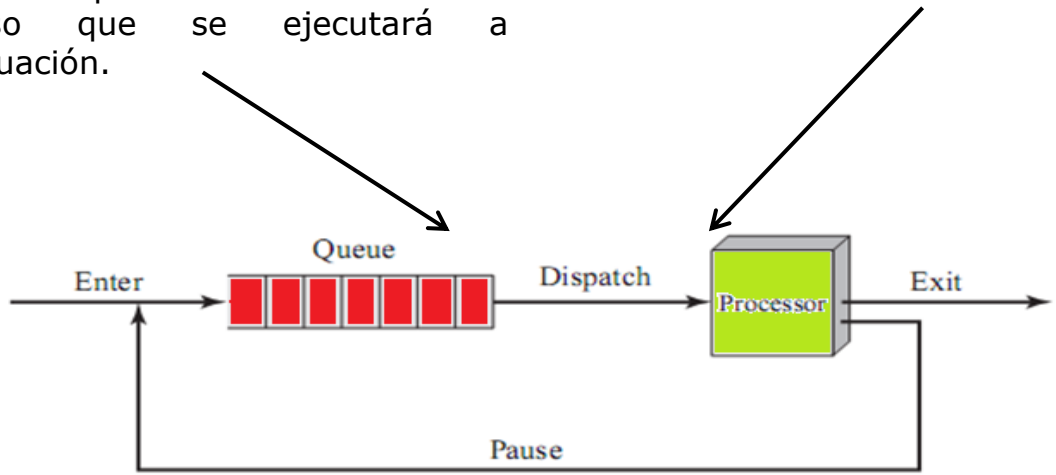
c) Listo. Puede entrar en ejecución en cuanto lo considere oportuno el planificador (módulo del SO que decide qué proceso pasará a ejecutarse). El estado del proceso reside en su BCP

Estado de los procesos. Un proceso puede encontrarse en diferentes situaciones, que cambian a medida que se modifican sus necesidades. Los estados básicos en que pueden encontrarse un proceso son: listo, ejecución y bloqueado



Planificador (scheduler). Forma parte del núcleo del SO. Entra en ejecución cada vez que se activa el SO y tiene por misión seleccionar el proceso que se ejecutará a continuación.

Activador (dispatcher). También forma parte del SO y se encarga de poner en ejecución el proceso seleccionado por el planificador.

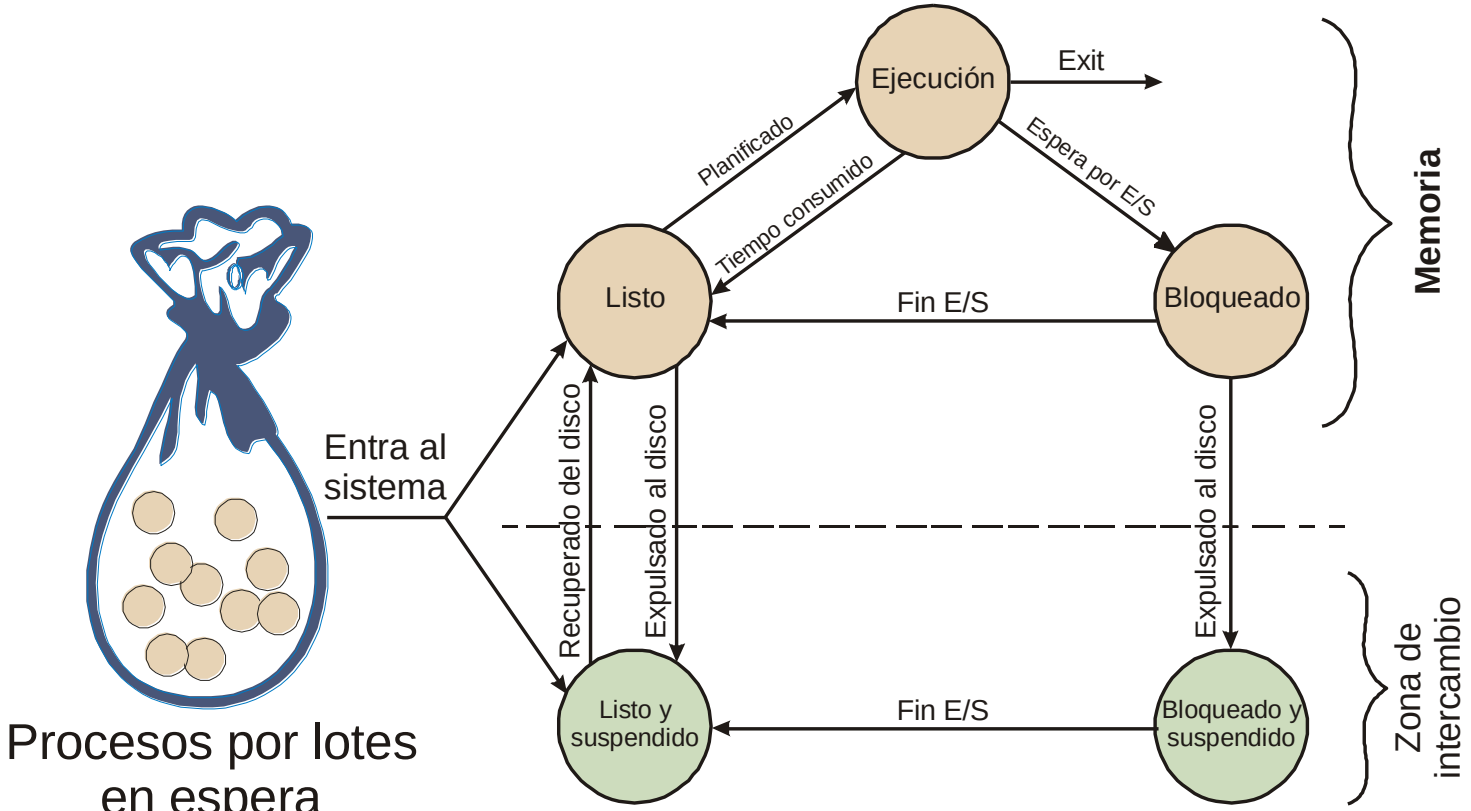


Estado de los procesos

Además de los estados básicos también pueden estar:

a) **Suspendido**. Para disminuir el grado de multiprogramación efectivo el SO puede retirar a un proceso sus marcos de página, alojándolo en la zona de intercambio (usualmente disco). El objetivo de esta operación es dejar suficiente memoria para los procesos no suspendidos, de forma que el conjunto residente no provoque hiperpaginación

b) En **espera**. Los procesos entran en el sistema porque lo solicita otro proceso o al estar prevista su ejecución *batch*. Es usual disponer de una lista de procesos *batch* en espera, para ser ejecutados en cuanto se pueda. El SO analiza dicha lista y lanza la ejecución de los procesos a medida que dispone de los recursos necesarios.


Ejecución
Exit
Planificado
Tiempo consumido
Espera por E/S
Listo
Bloqueado
Fin E/S
Memoria
Entra al sistema
Recuperado del disco
Expulsado al disco
Expulsado al disco
Listo y suspendido
Fin E/S
Bloqueado y suspendido
Zona de intercambio
Procesos por lotes en espera

Cambio de contexto

El cambio de contexto implica:

Salvar el estado del procesador en el BCP