UAI Universidad Abierta Interamericana

SISTEMAS OPERATIVOS

Mg. Leandro Ezequiel Mascarello

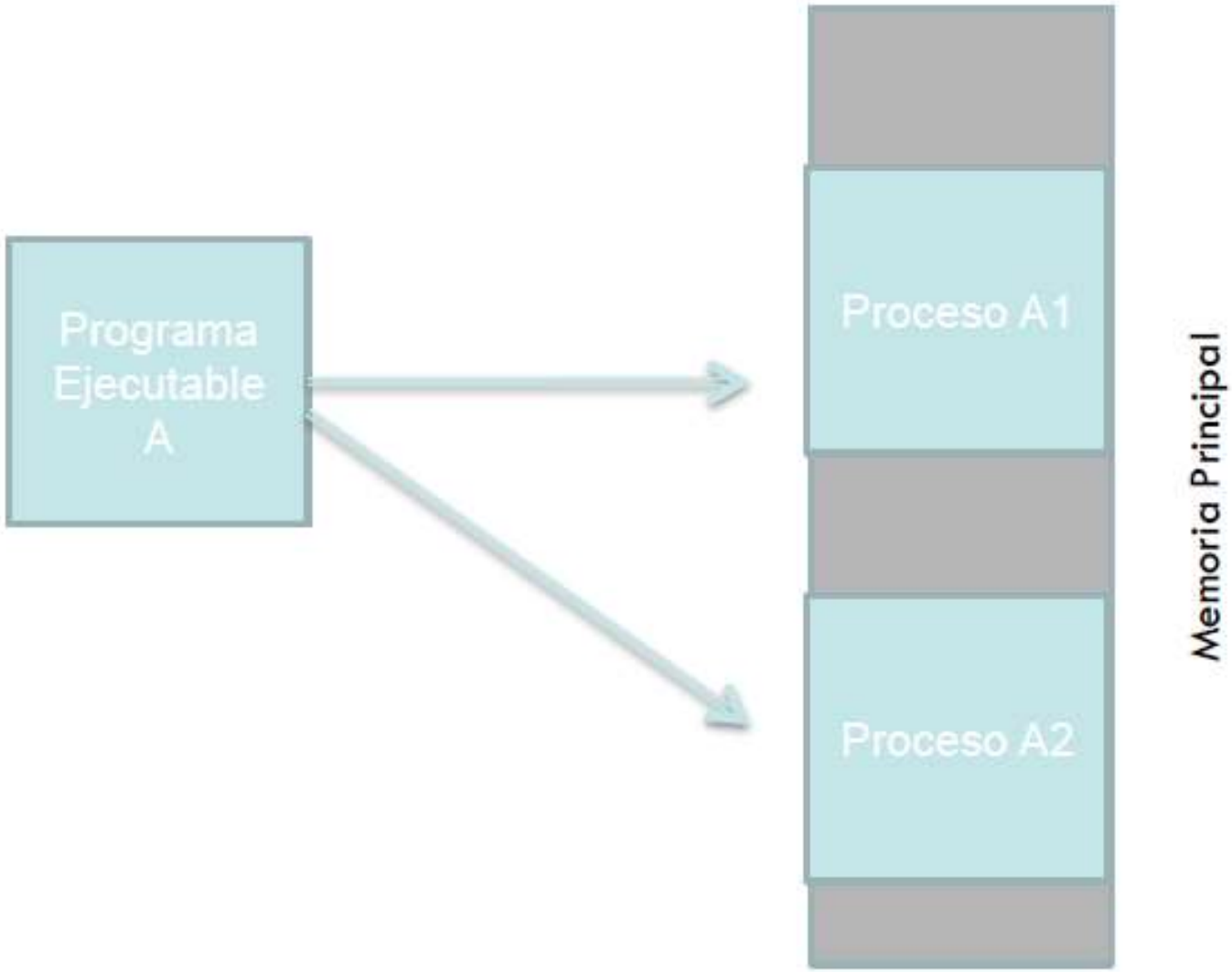
<leandro.mascarello@uai.edu.ar>

UAIOnline ultra

- Proceso: Instancia de unPrograma en ejecución.
  – Cada ejecución de un programa da lugar a un proceso.
  – El proceso → unidad de procesamiento que gestiona el sistema operativo.

- Un proceso está formado por:
  – Código del programa: Instrucciones.
  – Conjunto de datos asociados a la ejecución del programa

Concepto de Proceso

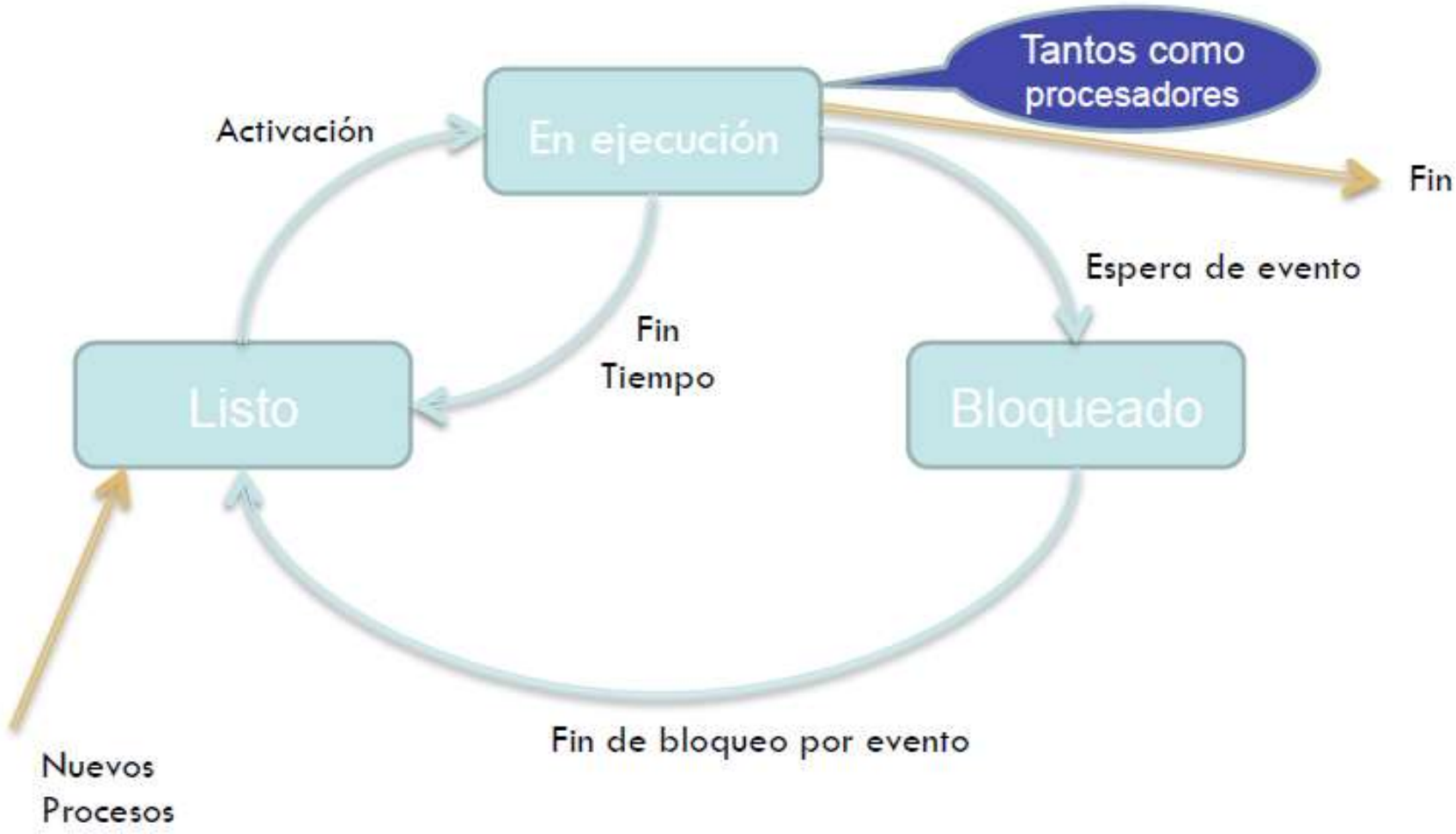


Representación en memoria

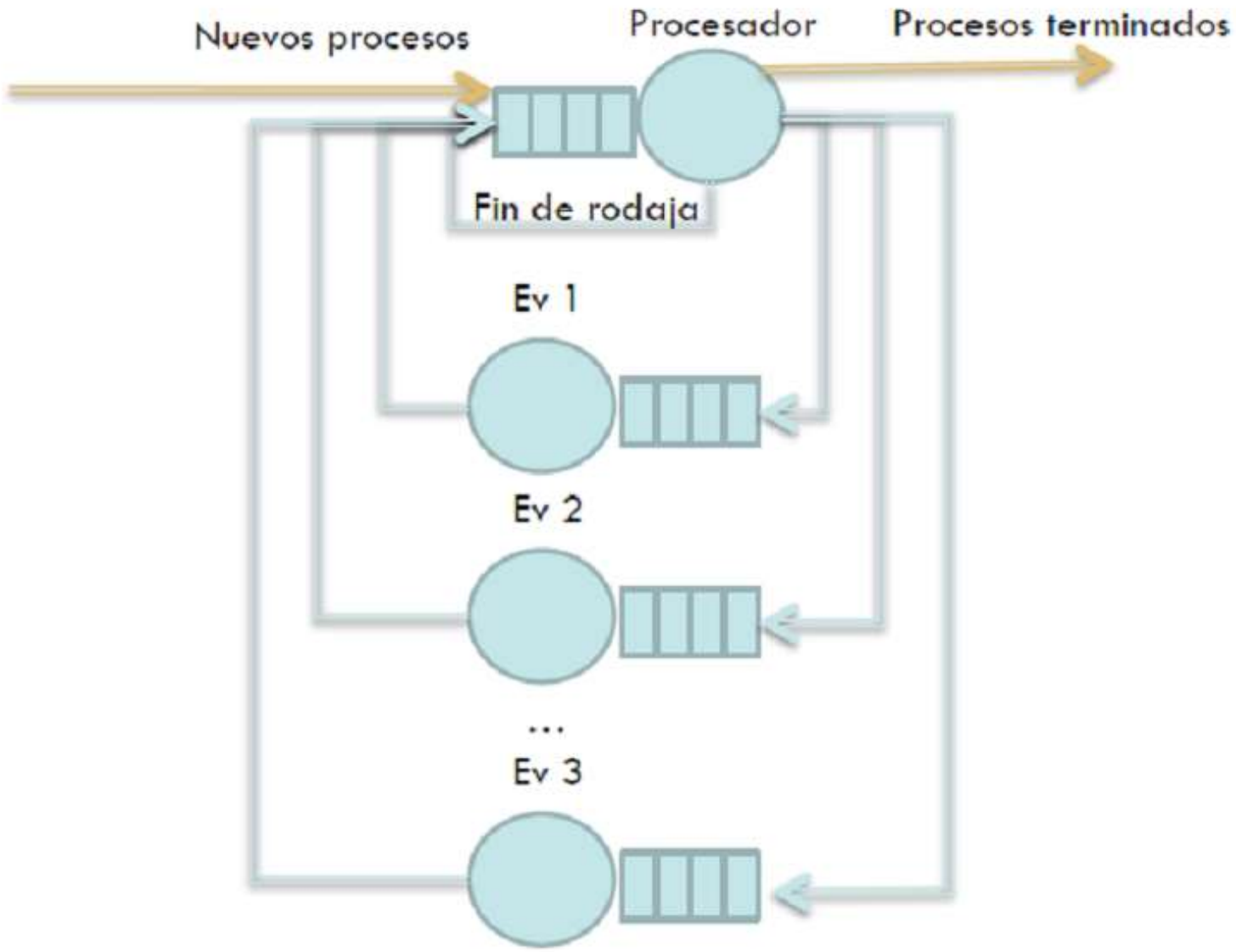
- Un proceso necesita memoria para las instrucciones y los datos.

- Distintas instancias de un programa necesitan zonas independientes para los datos.

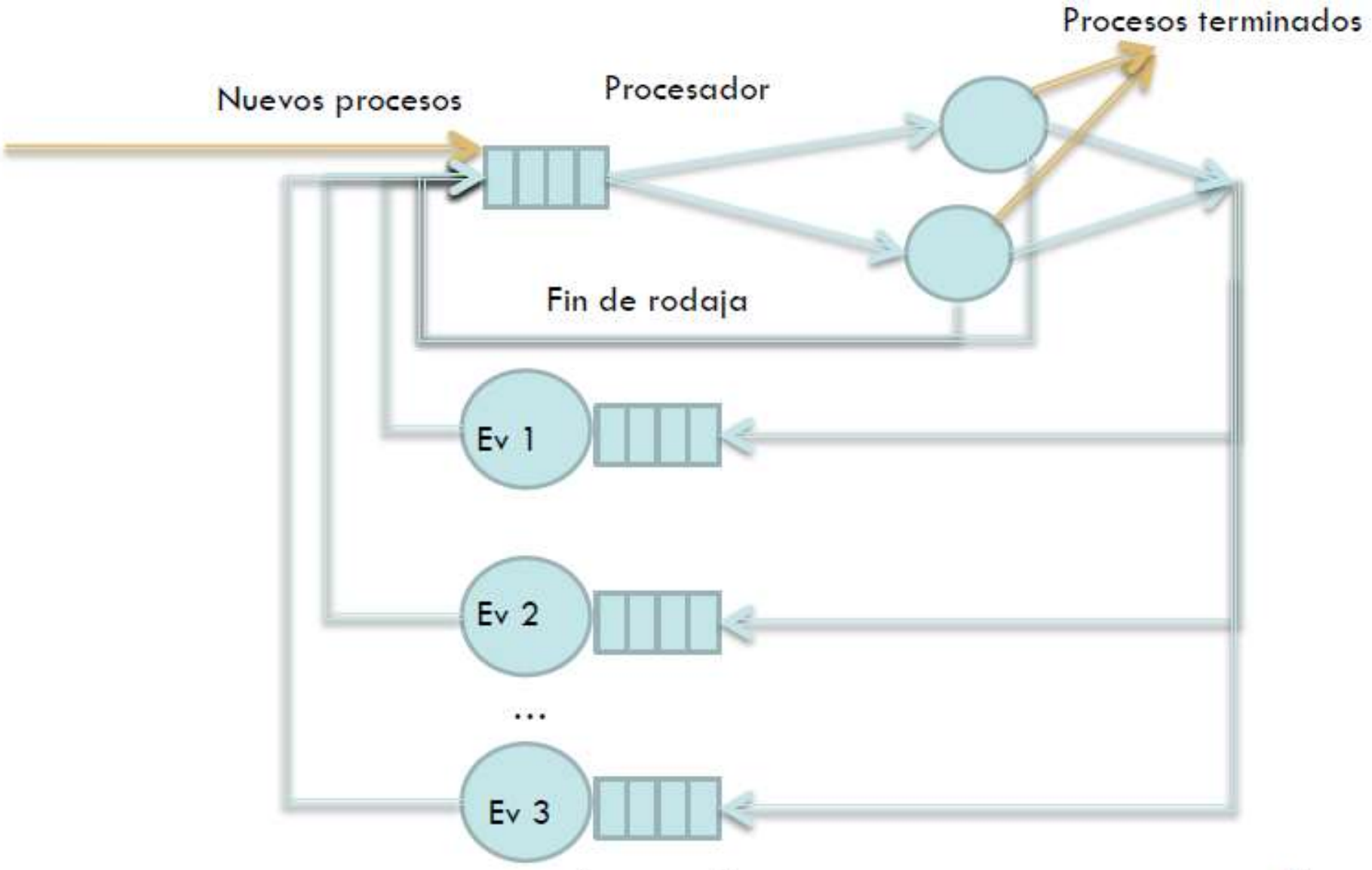
Ciclo de vida básico de un proceso



Modelo de colas simplificado: Un procesador



Modelo de colas simplificado: Varios procesadores

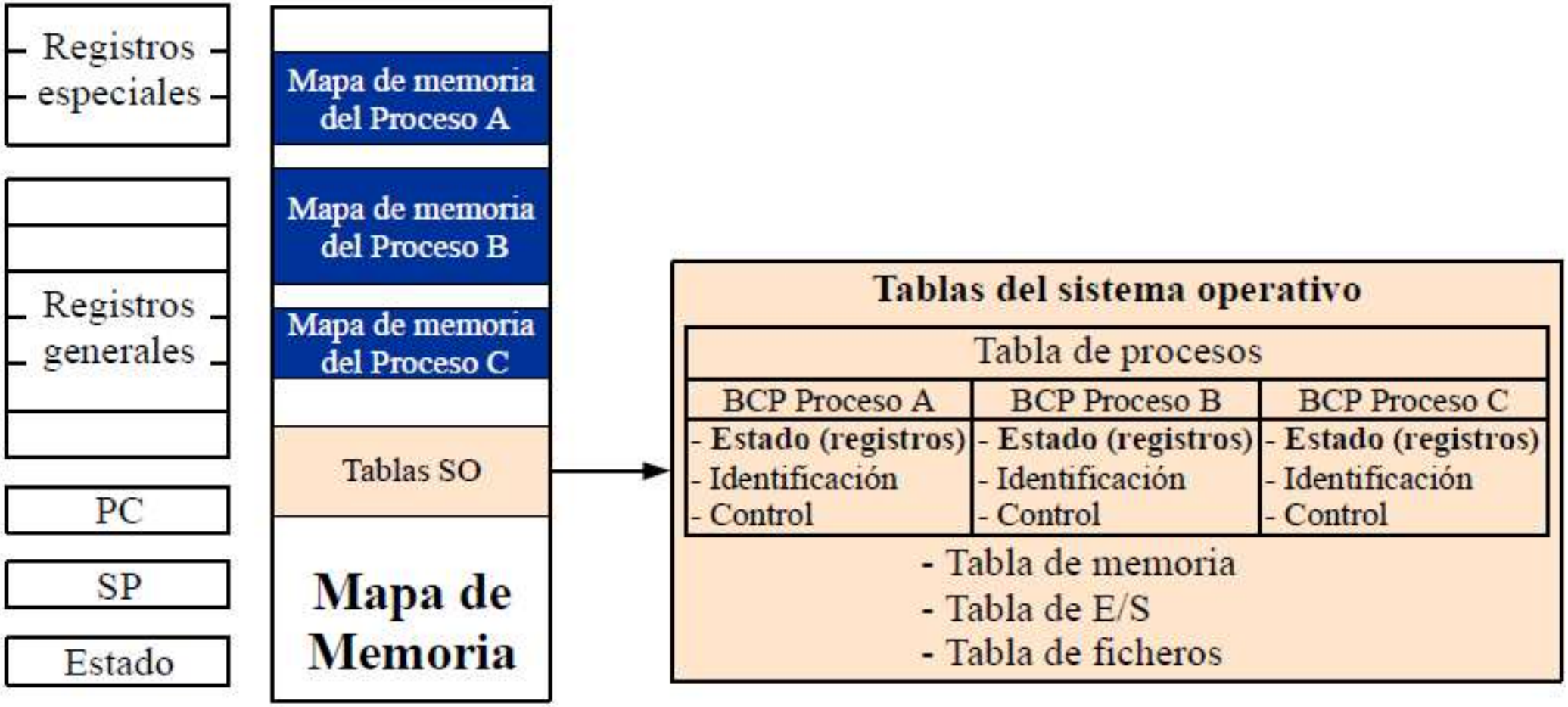


Información del proceso

- Toda la información que permite la correcta ejecución del proceso.

- Tres categorías:
  - Información almacenada en el procesador.
  - Información almacenada en memoria.
  - Información adicional gestionada por el sistema operativo.

Información del proceso



Estado del procesador

- El estado del procesador incluye los valores de los registros del procesador.
  - Registros accesibles en modo usuario.
    - Registros generales: Bancos de registros.
    - Contador de programa.
    - Puntero de pila.
    - Parte de usuario del registro de estado.
  - Registros accesibles en modo privilegiado:
  - Parte privilegiada del registro de estado.
  - Registros de control de memoria (p.ej. RBTP).
- Cambio de contexto:
  - Salvaguardar estado del procesador de proceso saliente.
  - Restaurar estado del procesador de proceso entrante.

Imagen de memoria de un proceso

- La imagen de memoria está formada por los **espacios de memoria** que un proceso está autorizado a utilizar.

- Si un proceso genera una dirección que esta fuera del espacio de direcciones el HW genera un **trap**.

- La imagen de memoria dependiendo del computador puede estar referida a memoria virtual o memoria física.

Modelos de imagen de memoria: Región única

- Proceso con única región de tamaño fijo.
  - Usado en sistemas sin memoria virtual.

- Proceso con única región de tamaño variable.
  - Sistemas sin memoria virtual:
    - Necesita espacio de reserva → Desperdicio de memoria.
  - Sistemas con memoria virtual:
    - Espacio de reserva virtual → Factible pero menos flexible que múltiples regiones.

Modelos de imagen de memoria: Regiones múltiples

- Proceso con número fijo de regiones de tamaño variable.

– Regiones prefijadas (texto, datos, pila).

– Cada región puede crecer.

– Con memoria virtual el hueco entre pila y datos no consume recursos físicos.

Modelos de imagen de memoria: Región única

- Proceso con un número variable de regiones de tamaño variable.

– Opción más avanzada (usada en versiones actuales de Windows y UNIX).

– Un proceso se estructura en un número arbitrario de regiones.

– Muy flexible:

  - Regiones compartidas.
  - Regiones con distintos permisos.

Información del sistema operativo

- El sistema operativo mantiene información adicional sobre los procesos.
  - El sistema operativo mantiene esta información en una tabla: **Tabla de Procesos**.
  - **Bloque de control de Procesos** (BCP): Cada entrada de la tabla que mantiene la información sobre un proceso.
  - En el BCP se mantiene casi toda la información sobre un proceso.
    - Algunos elementos de información se mantienen fuera por motivos de implementación.

Contenidos del BCP

- Información de identificación.
- Estado del procesador.
- Información de control del proceso.

Ejemplo:

- Identificador del proceso.
- Identificador del proceso padre.
- Información sobre el usuario.

Contenidos del BCP

- Información de identificación.
- Estado del procesador.
- Información de control del proceso.

Información de planificación y estado:
- Estado del proceso.
- Evento por el que espera (si bloqueado)
- Prioridad del proceso.
- Información de planificación.

Descripción de regiones asignada.
Recursos asignados:
- Archivos abiertos.
- Puertos de comunicaciones usados.
- Temporizadores.

Punteros para estructurar los procesos en colas (o anillos).
Información para comunicación entre procesos.

Contenidos del BCP

- Información de identificación.
- Estado del procesador.
- Información de control del proceso.

Al iniciar el proceso:
- Valores iniciales para el estado del procesador.

Después de un cambio de contexto:
- Copia de los valores del estado del procesador.

Información fuera del BCP

- No toda la información referida a un proceso se almacena en el BCP. Se decide qué almacenar en función de:
  - La eficiencia.
    - Las tablas pueden tener un tamaño predefinido y siempre está residente en memoria.
    - Hay que optimizar su tamaño.
  - Compartir información
    - Si hay que compartir algún dato éste no puede estar en el BCP.
    - Se usan punteros para apuntar a otras estructuras, otras tablas, permitiéndose así el compartir información:
      - Ficheros abiertos.
      - Páginas de memoria.

- Se sitúan fuera del BCP.
- Describe la imagen de memoria del proceso
- El BCP contiene el puntero a la tabla de páginas
- Razones:
  - Tiene tamaño variable
  - La compartición de memoria entre procesos requiere que sea externa al BCP

Punteros de posición de los ficheros

- Se sitúan fuera del BCP.

- Si se añaden a la tabla de ficheros abiertos (en el BCP) no se pueden compartir.

- Si se asocian al nodo-i se comparten siempre.

- Se ponen en una estructura común a los procesos y se asigna uno nuevo en cada servicio OPEN.

Ejemplo: Ejecución de un mandato


```
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char** argv) {
pid_t pid;
pid = fork();                                   prog cat f1
switch (pid) {
        case -1: /* error */
                exit(-1);
        case 0: /* proceso hijo */
                if (execvp(argv[1], &argv[1])<0) {
                perror("error"); }
                break;
        default:
                printf("Proceso padre");
        }
return 0;
}
```

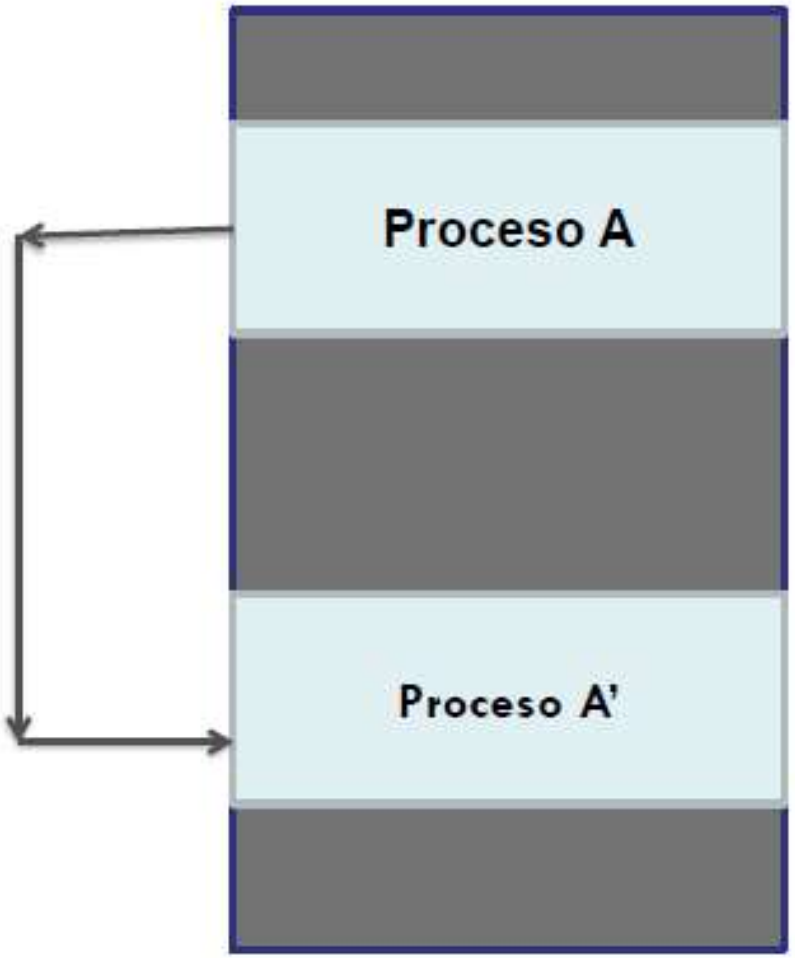
Servicio fork

- `pid_t fork(void);`

- Duplica el proceso que invoca la llamada.
- El proceso padre y el proceso hijo siguen ejecutando el mismo programa.
- El proceso hijo hereda los ficheros abiertos del proceso padre.
  - Se copian los descriptores de archivos abiertos.
- Se desactivan las alarmas pendientes.

- Devuelve:
  - -1 el caso de error.
  - En el proceso padre: el identificador del proceso hijo.
  - En el proceso hijo: 0

Concepto de Proceso



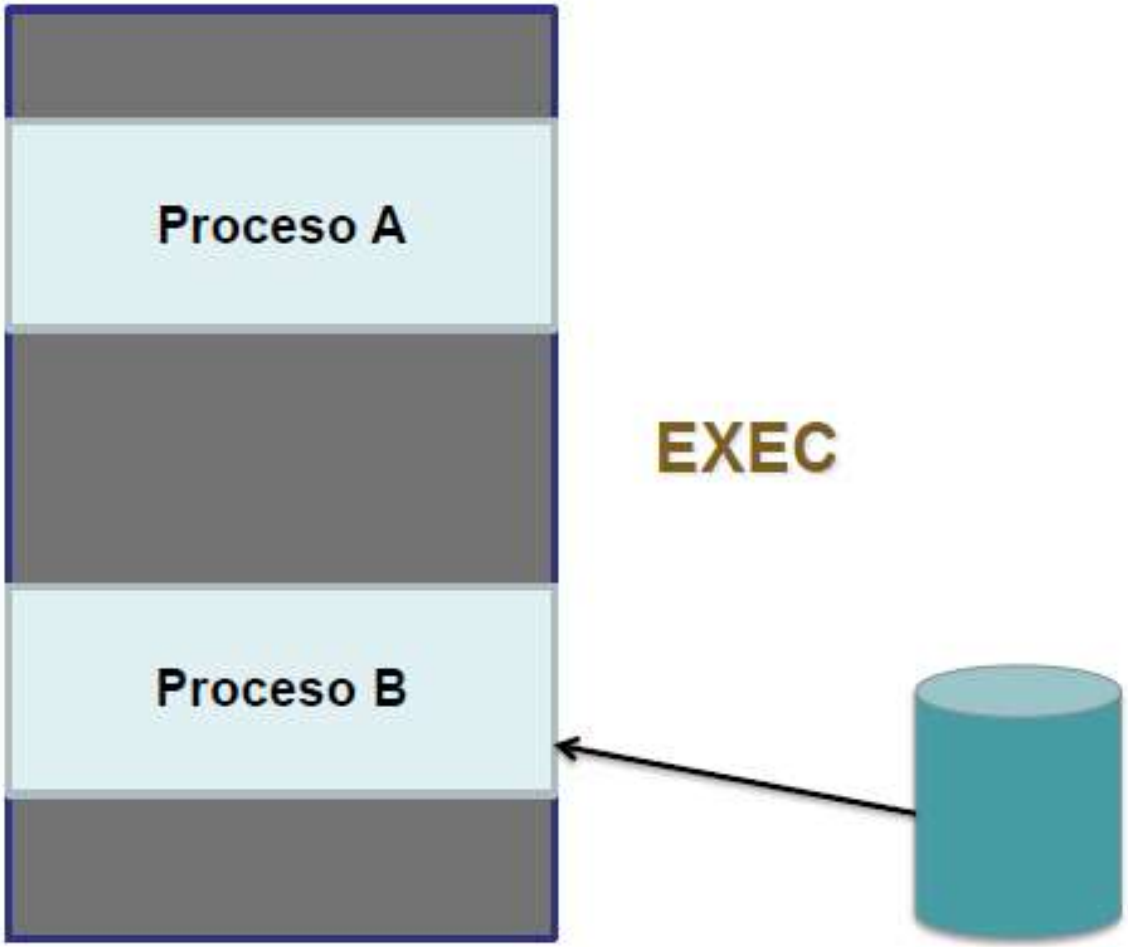
FORK

Servicio exec

- Servicio único pero múltiples funciones de biblioteca.
- ```
  int execl(const char *path, const char *arg, ...);
  int execv(const char* path, char* const argv[]);
  int execve(const char* path, char* const argv[], char* const envp[]);
  int execvp(const char *file, char *const argv[])
  ```

- Cambia la imagen del proceso actual.
  - path: Ruta al archivo ejecutable.
  - file: Busca el archivo ejecutable en todos los directorios especificados por PATH.
- Descripción:
  - Devuelve -1 en caso de error, en caso contrario no retorna.
  - El mismo proceso ejecuta otro programa.
  - Los ficheros abiertos permanecen abiertos.
  - Las señales con la acción por defecto seguirán por defecto, las señales con manejador tomarán la acción por defecto.
  ```

Servicio Fork

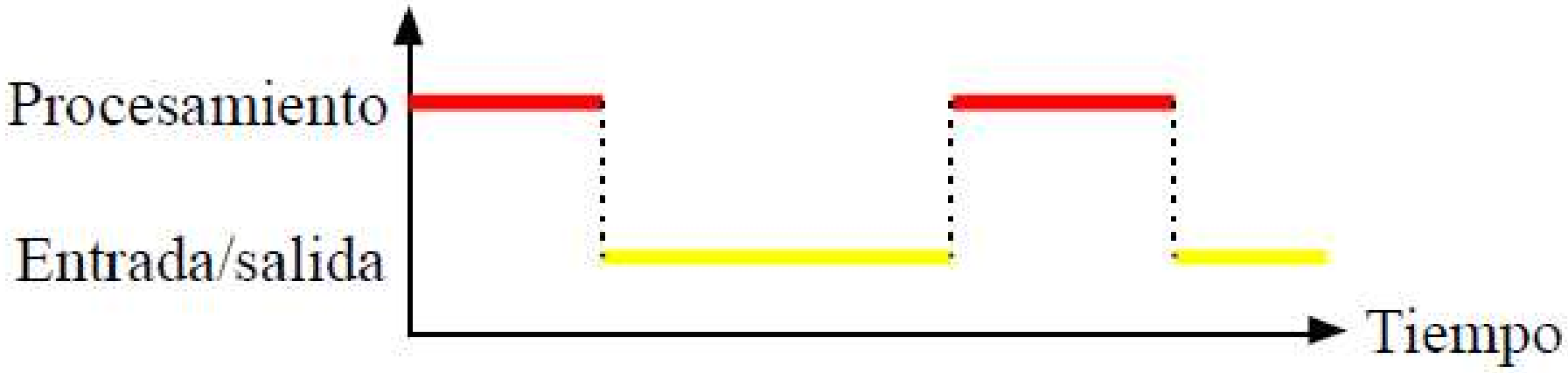


Servicio exit

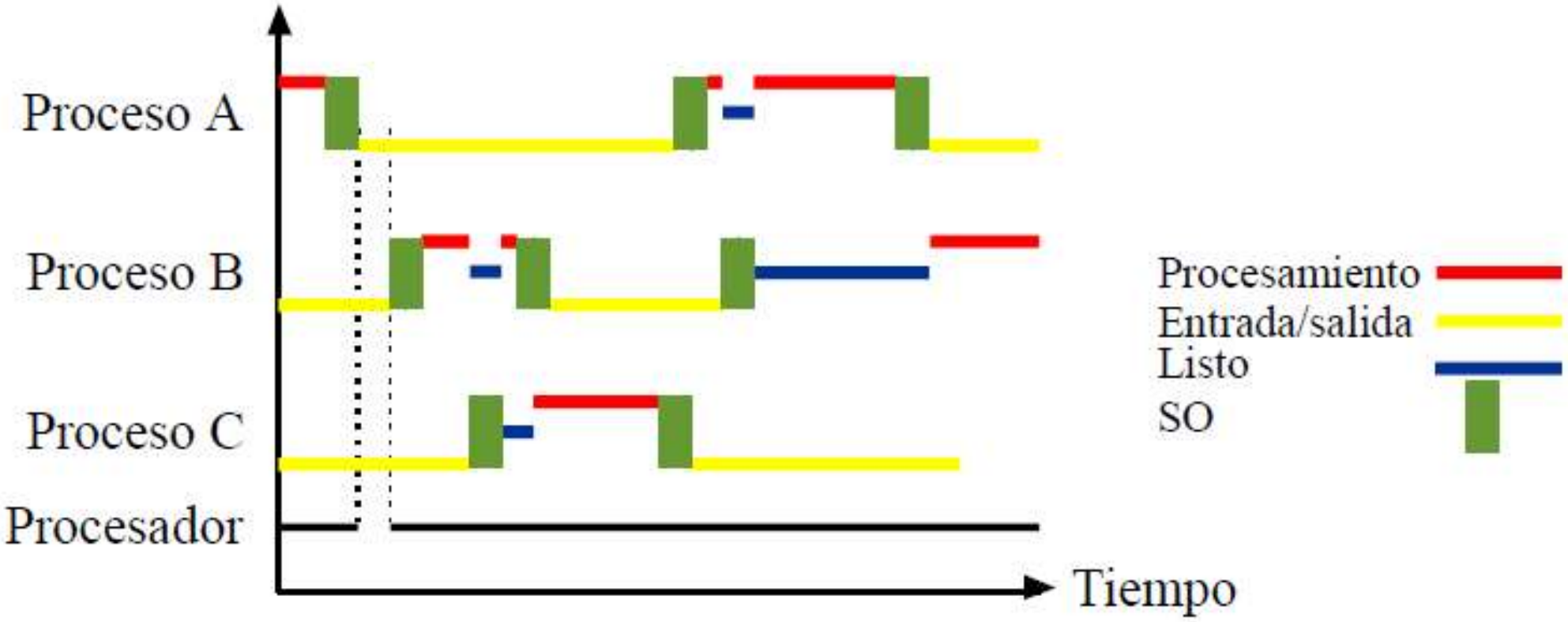
- Finaliza la ejecución del proceso.
- **void exit(status);**
- Se cierran todos los descriptores de ficheros abiertos.
- Se liberan todos los recursos del proceso.
- Se libera el BCP del proceso.

Tipos de sistemas operativos

- Paralelismo real entre E/S y UCP (DMA)
- Alternancia en los procesos de fases de E/S y de procesamiento
- La memoria almacena varios procesos



Ejecución en un sistema multitarea



Ventajas de la multitarea

- Facilita la programación, dividiendo los programas en procesos (modularidad).

- Permite el servicio interactivo simultáneo de varios usuarios de forma eficiente.

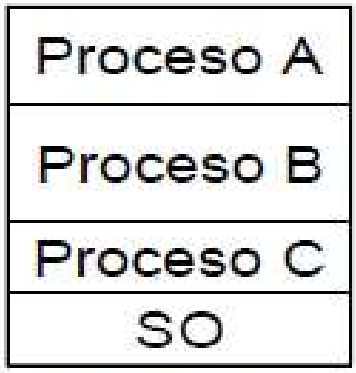
- Aprovecha los tiempos que los procesos pasan esperando a que se completen sus operaciones de E/S.

- Aumenta el uso de la CPU.

Grado de multiprogramación

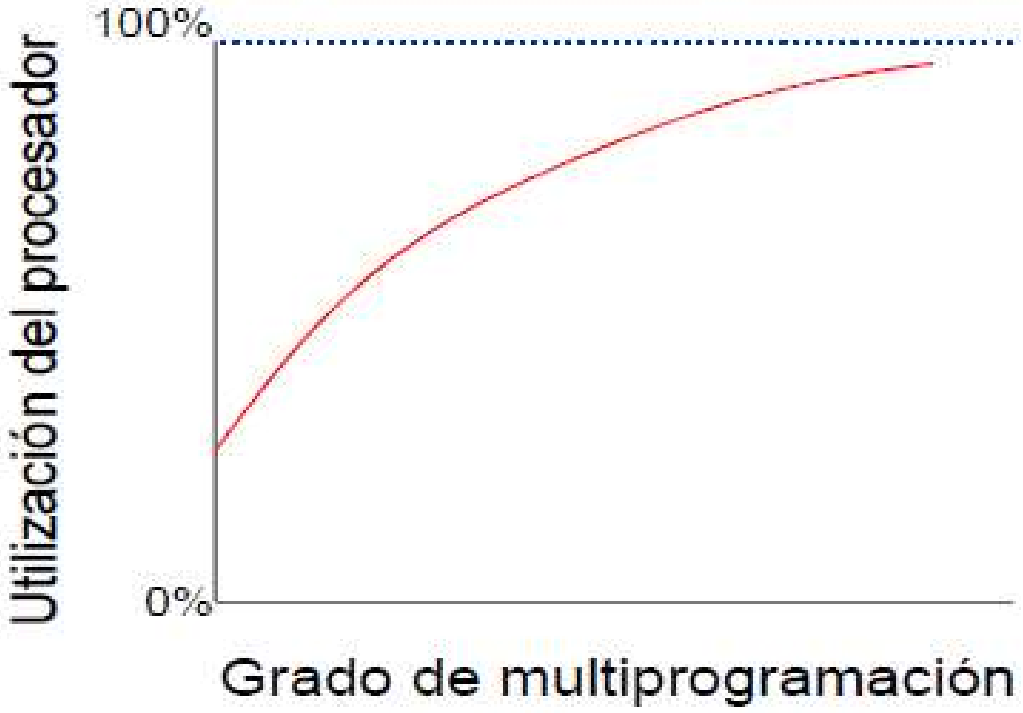
- Grado de multiprogramación: nº de procesos activos

- Necesidades de memoria principal: Sistema sin memoria virtual

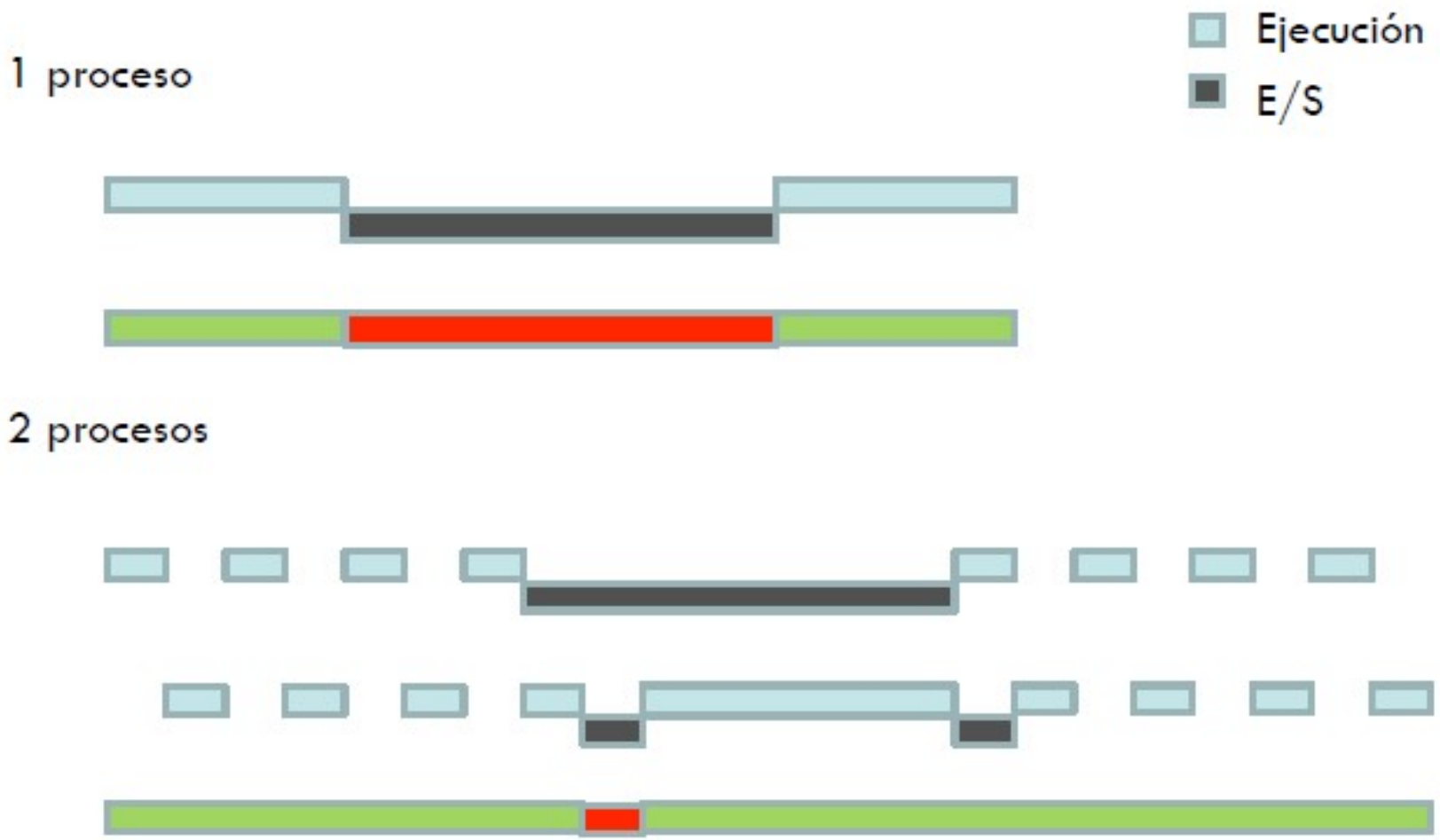


Memoria principal

Cada proceso reside totalmente en M.p



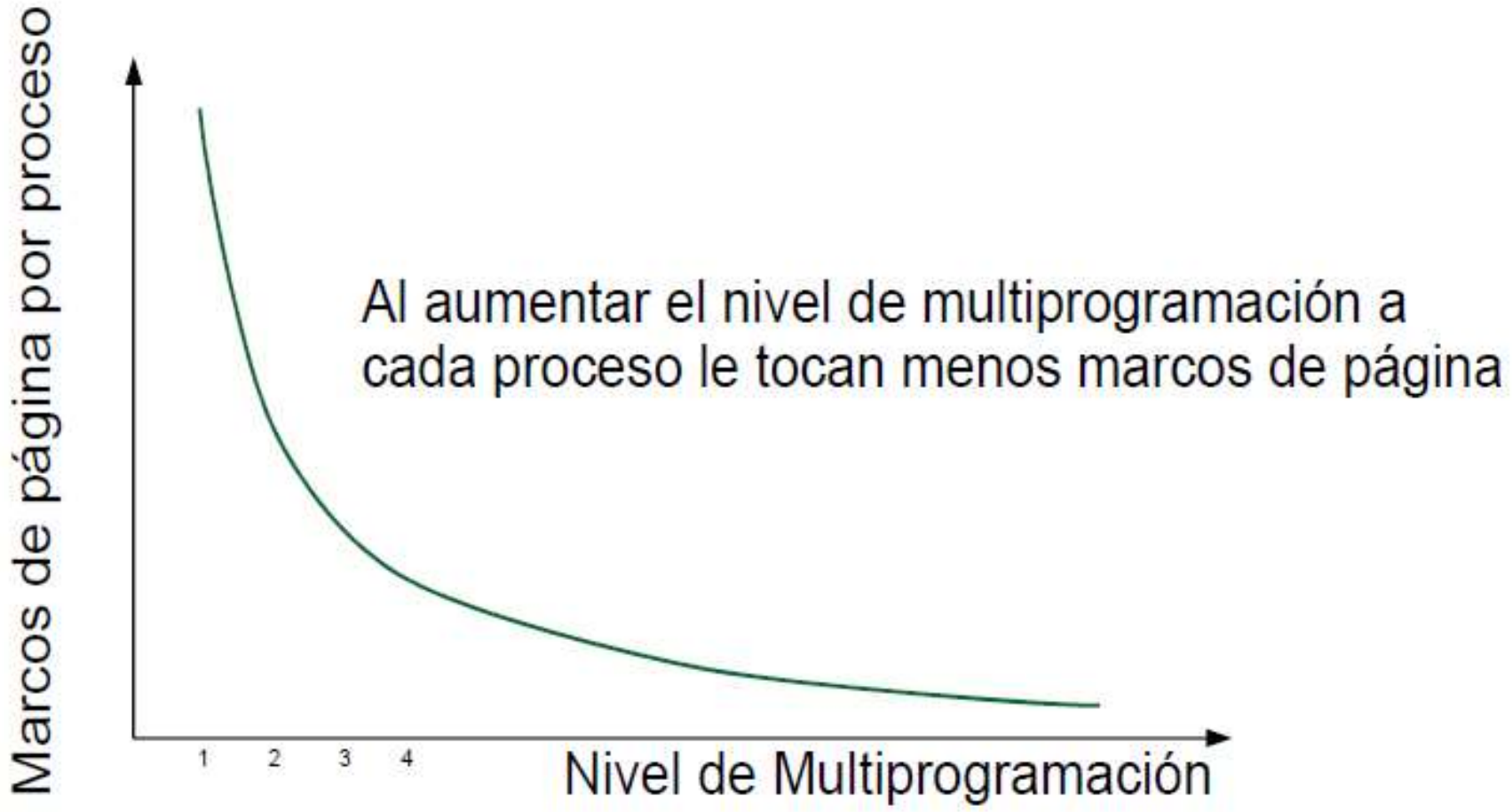
Multiprogramación: uso de la CPU



- Los sistemas con memoria virtual:
  – Dividen el espacio direccionable de los procesos en páginas.
  – Dividen el espacio direccionable de la memoria física principal en marcos de página.

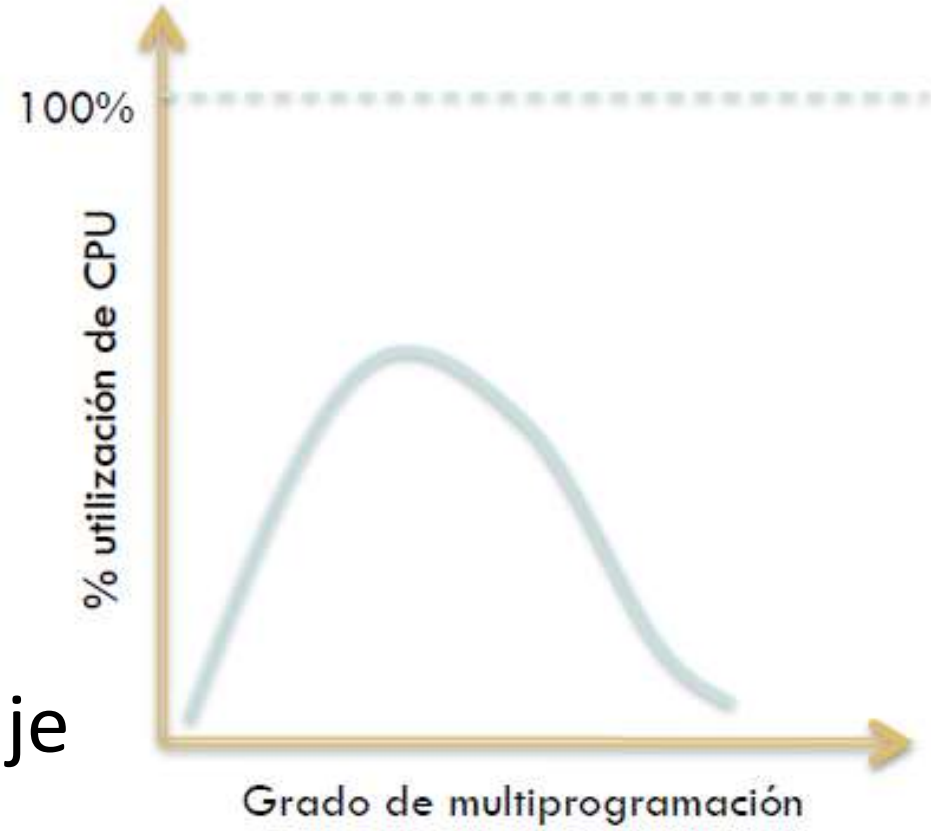
- En un momento dado cada proceso tiene un cierto número de sus páginas en memoria principal (conjunto residente).

Necesidad de memoria: Sistema con memoria virtual



Rendimiento: Poca memoria física

- Al aumentar el grado de multiprogramación:
  - Desciende el tamaño del conjunto residente de cada proceso.



- Se produce Hiperpaginación antes de alcanzar un porcentaje alto de uso de CPU.

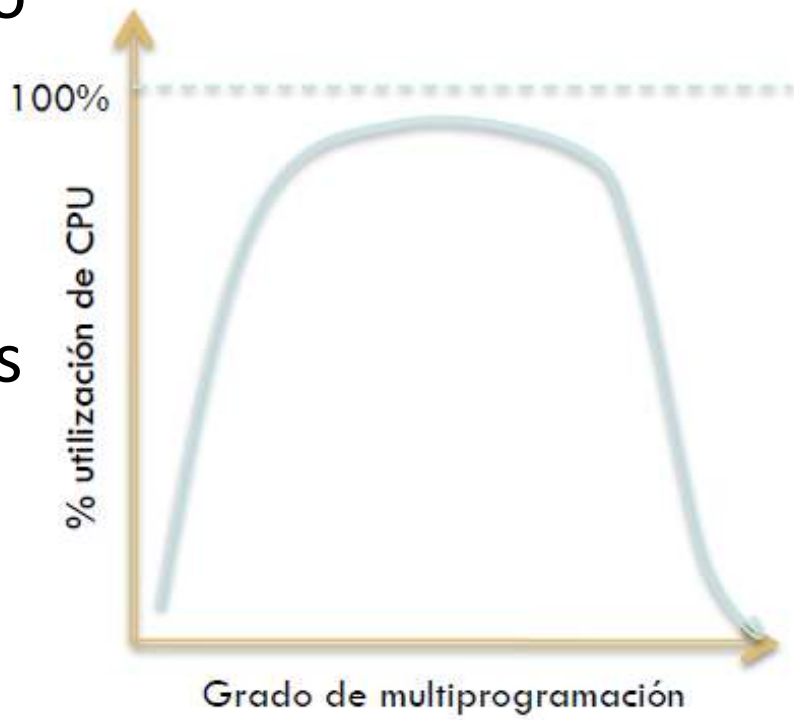
- **Solución**: Ampliación de memoria principal.

Rendimiento: Mucha memoria física

- Al aumentar el grado de multiprogramación:
  - Desciende el tamaño del conjunto residente de cada proceso.

- Se alcanza un alto porcentaje de utilización de CPU con menos procesos de los que caben en memoria.

- **Solución**: Mejora del procesador o incorporación de más procesadores.

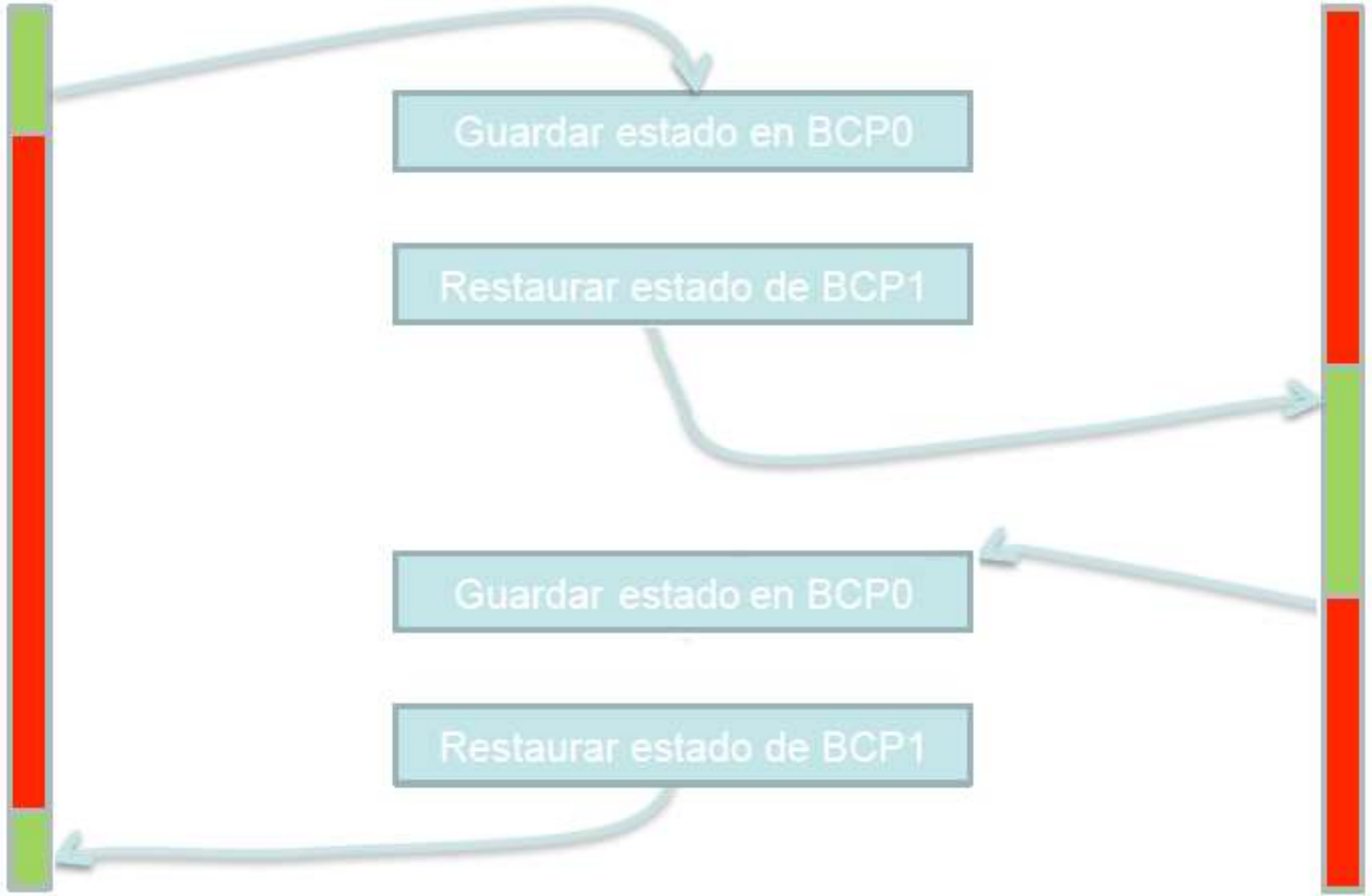


Cambios de contexto

- Se produce cuando el sistema operativo asigna el procesador a un nuevo proceso.

- Acciones:
  - Guardar el estado del procesador en el BCP del proceso en ejecución.
  - Restaurar el estado del nuevo proceso en el procesador.

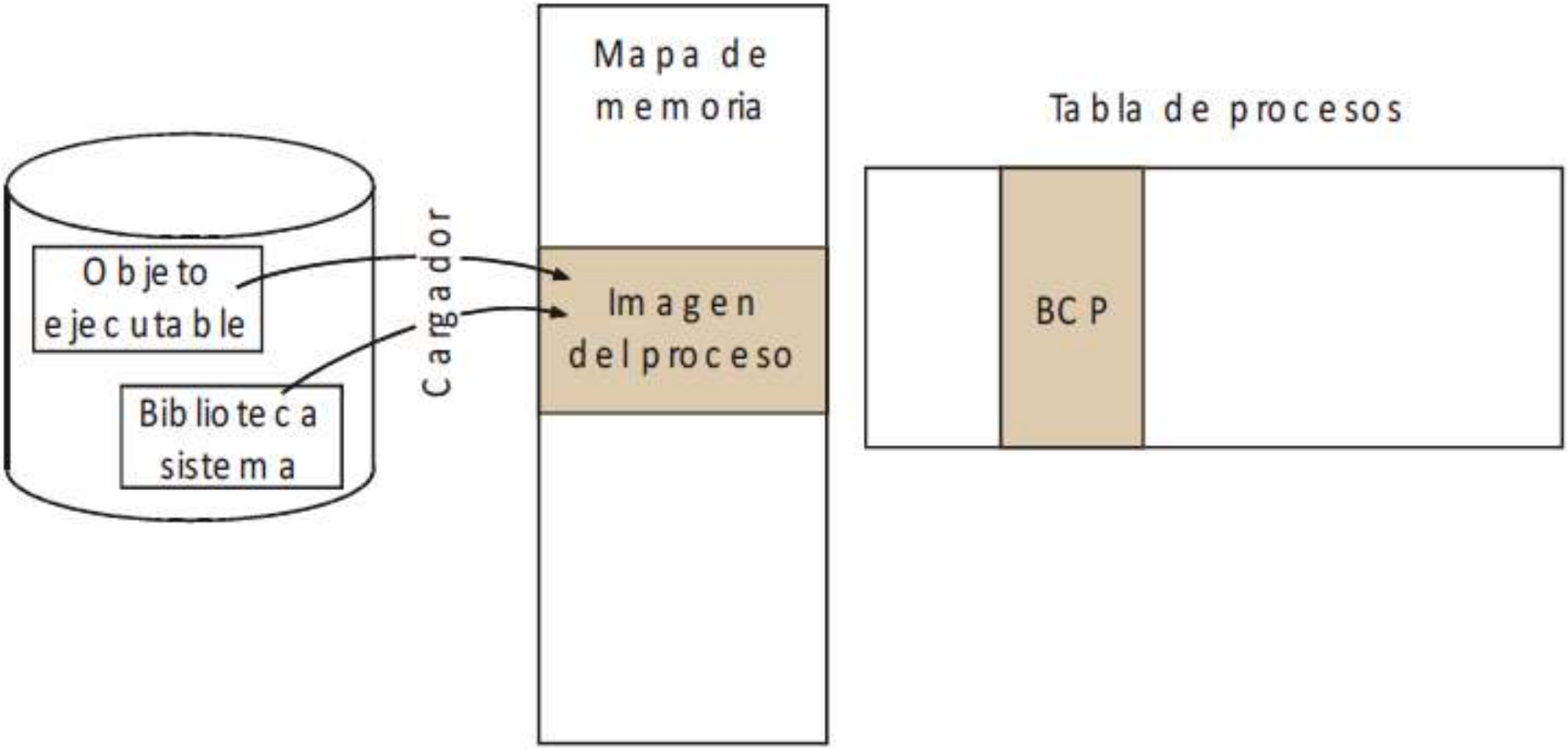
Cambio de contexto



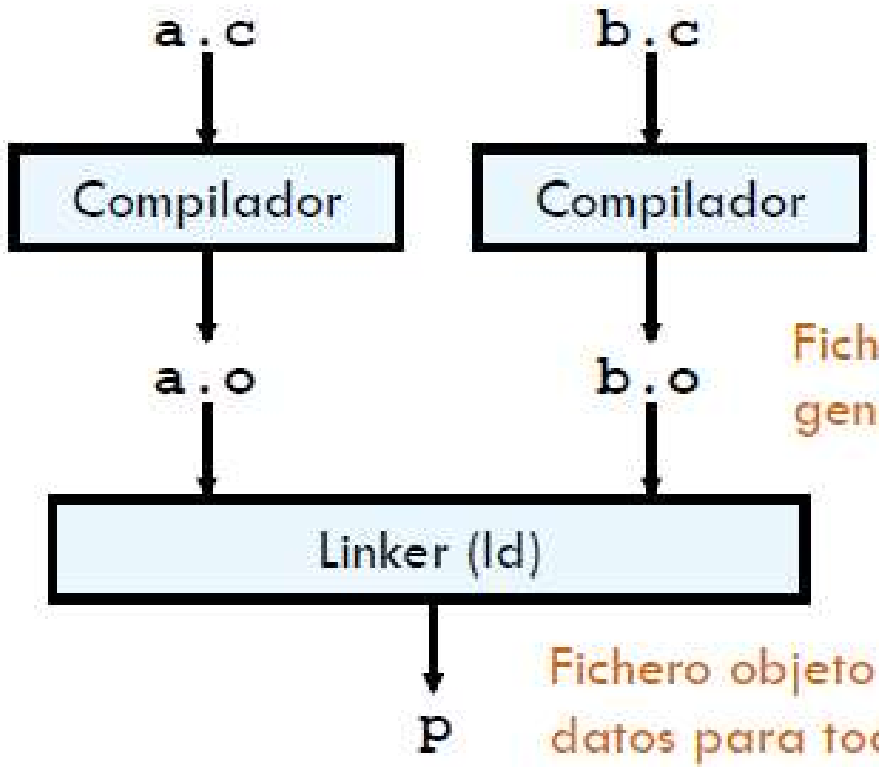
Tipos de cambio de contexto

- **Cambio de contexto *voluntario* (C.C.V):**
  - Proceso realiza llamada al sistema (o produce una excepción como un fallo de página) que implica esperar por un evento.
  - *en_ejecución* → *bloqueado.*
  - Ejemplos: leer del terminal, fallo de página.
  - ¿Motivo? ⇒ *Eficiencia en el uso del procesador*
- **Cambio de contexto *involuntario* (C.C.I):**
  - SO quita de la CPU al proceso
  - *En ejecución* → *listo*
  - Ejemplos: fin de rodaja de ejecución o pasa a *listo* proceso bloqueado de mayor prioridad
  - ¿Motivo? ⇒ *Reparto del uso del procesador*

Formación de un proceso



Generación de ejecutables



Ficheros objeto reubicables generados por separado

Fichero objeto ejecutable (contiene código y datos para todas las funciones definidas en a.c y b.c)

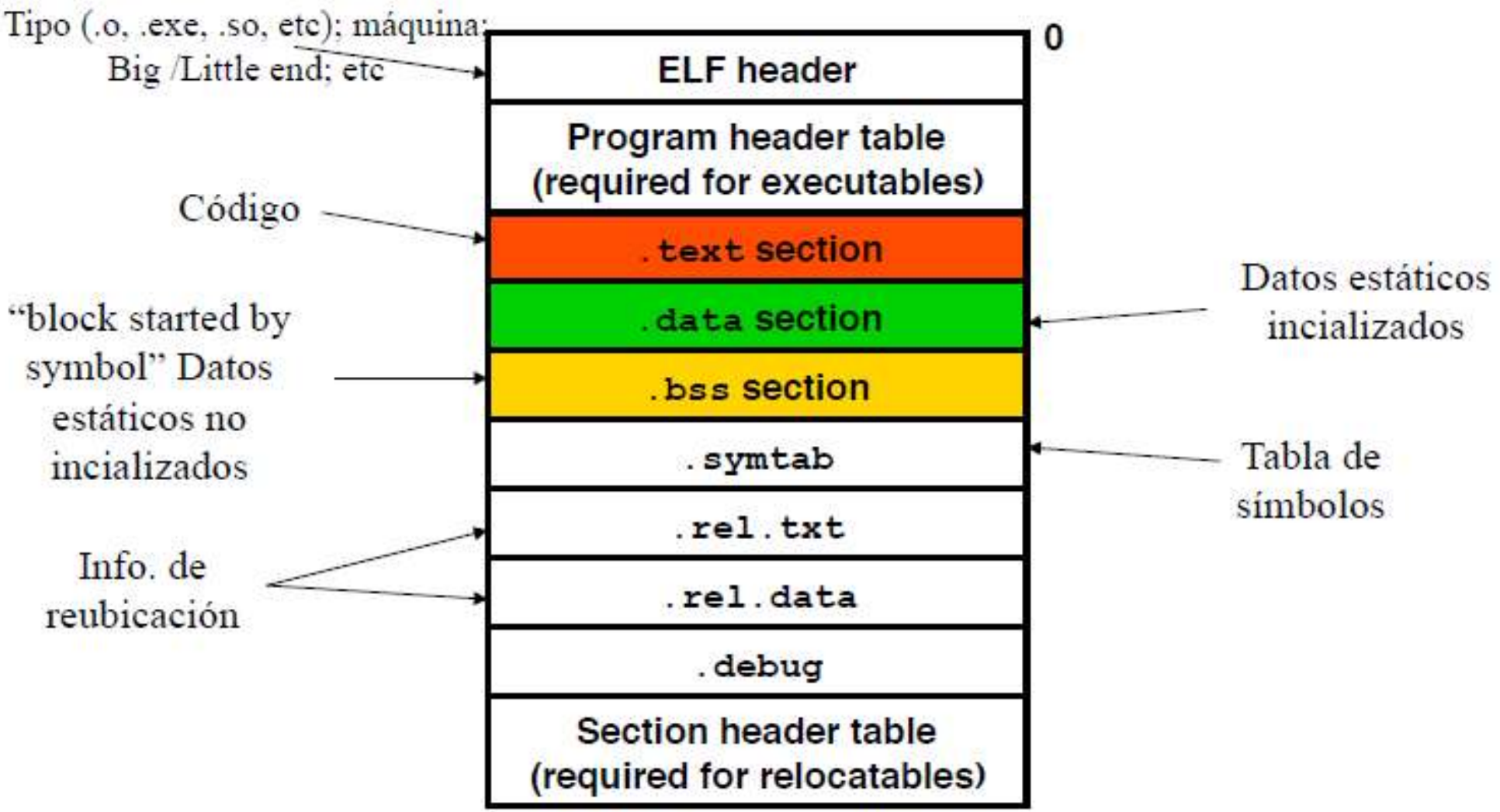
Editor de enlaces (linker)

- Combina los ficheros objeto:
  - fusiona los diferentes ficheros objeto reubicables (.o) en un único fichero objeto ejecutable: input del cargador
- Resuelve las referencias externas:
  - referencias a símbolos definidas en otro fichero objeto
- Reubica los símbolos:
  - de su posiciones relativas en los .o a las absolutas en el ejecutable: reajusta las refs a estas nuevas posiciones
  - símbolos: refs. de funciones (código) y de datos

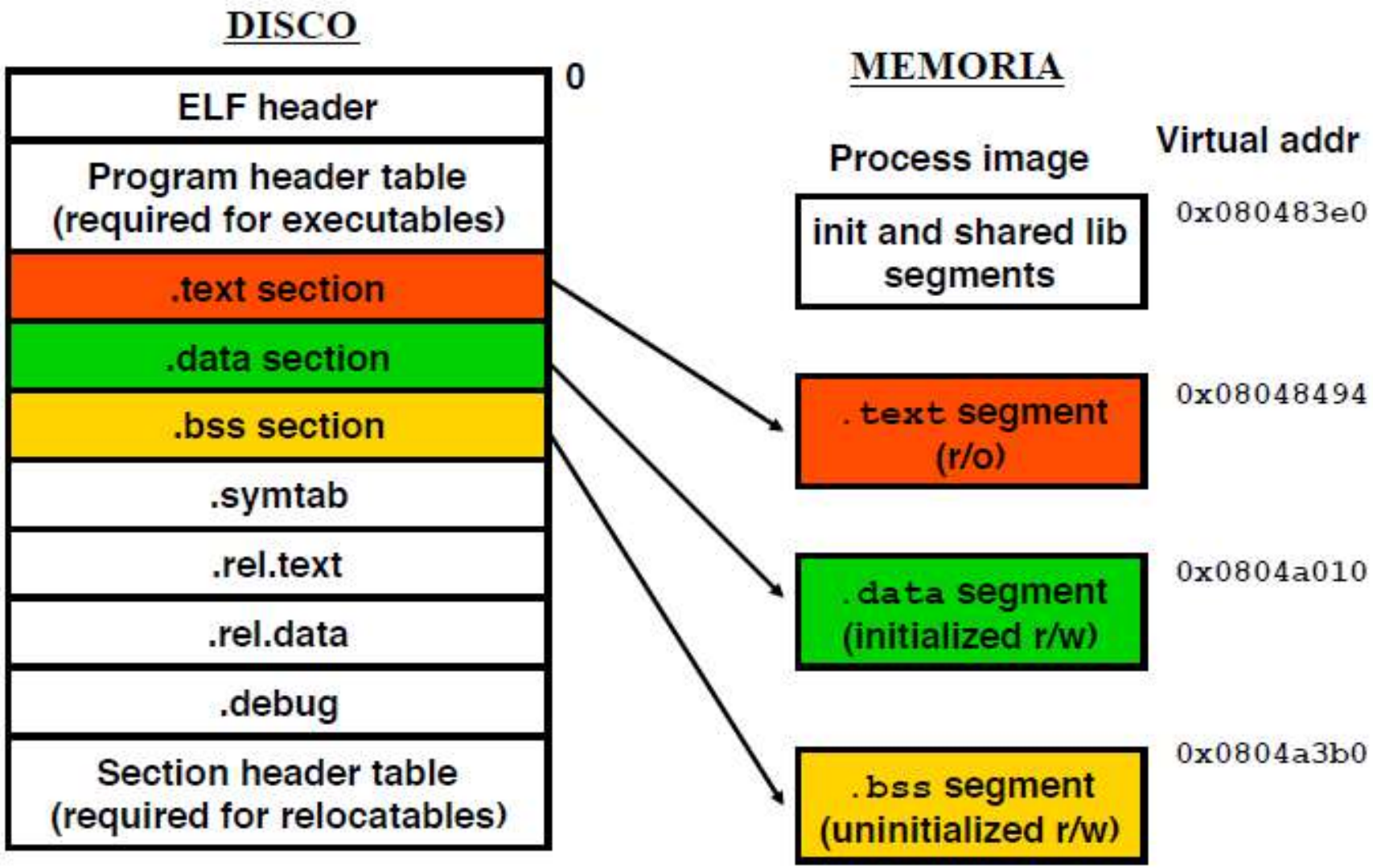
Ejemplo: Formato ELF

- ELF: Executable and Linkable Format
  - formato binario estándar para ficheros objeto
  - original de System V → BSD, Linux, Solaris
  - formato unificado para:
    - ficheros objeto reubicables
    - ficheros objeto ejecutables
    - ficheros objeto compartidos

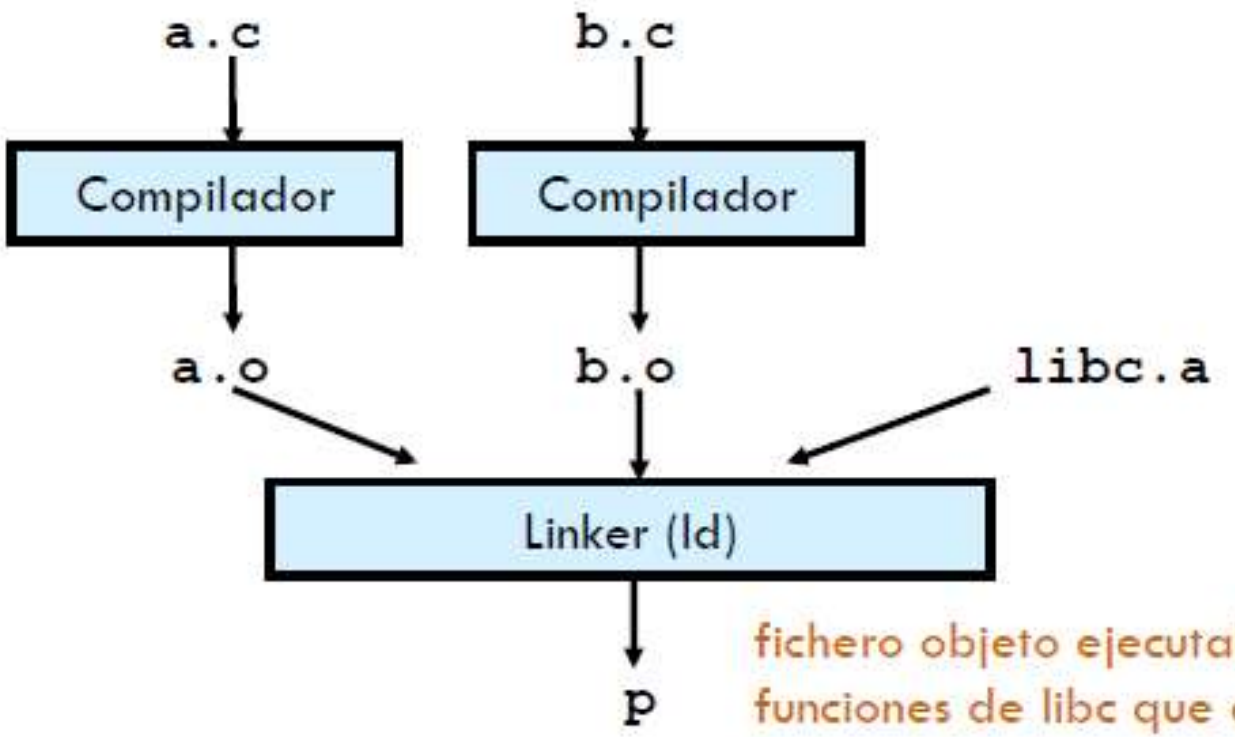
Formato ELF


Tipo (.o, .exe, .so, etc); máquina; Big /Little end; etc
0
ELF header
Program header table (required for executables)
Código
.text section
Datos estáticos incializados
.data section
"block started by symbol" Datos estáticos no incializados
.bss section
.symtab
Tabla de símbolos
.rel.txt
Info. de reubicación
.rel.data
.debug
Section header table (required for relocatables)

Carga de ejecutable



Bibliotecas estáticas



librería estática: fichero concatenando ficheros objeto reubicables

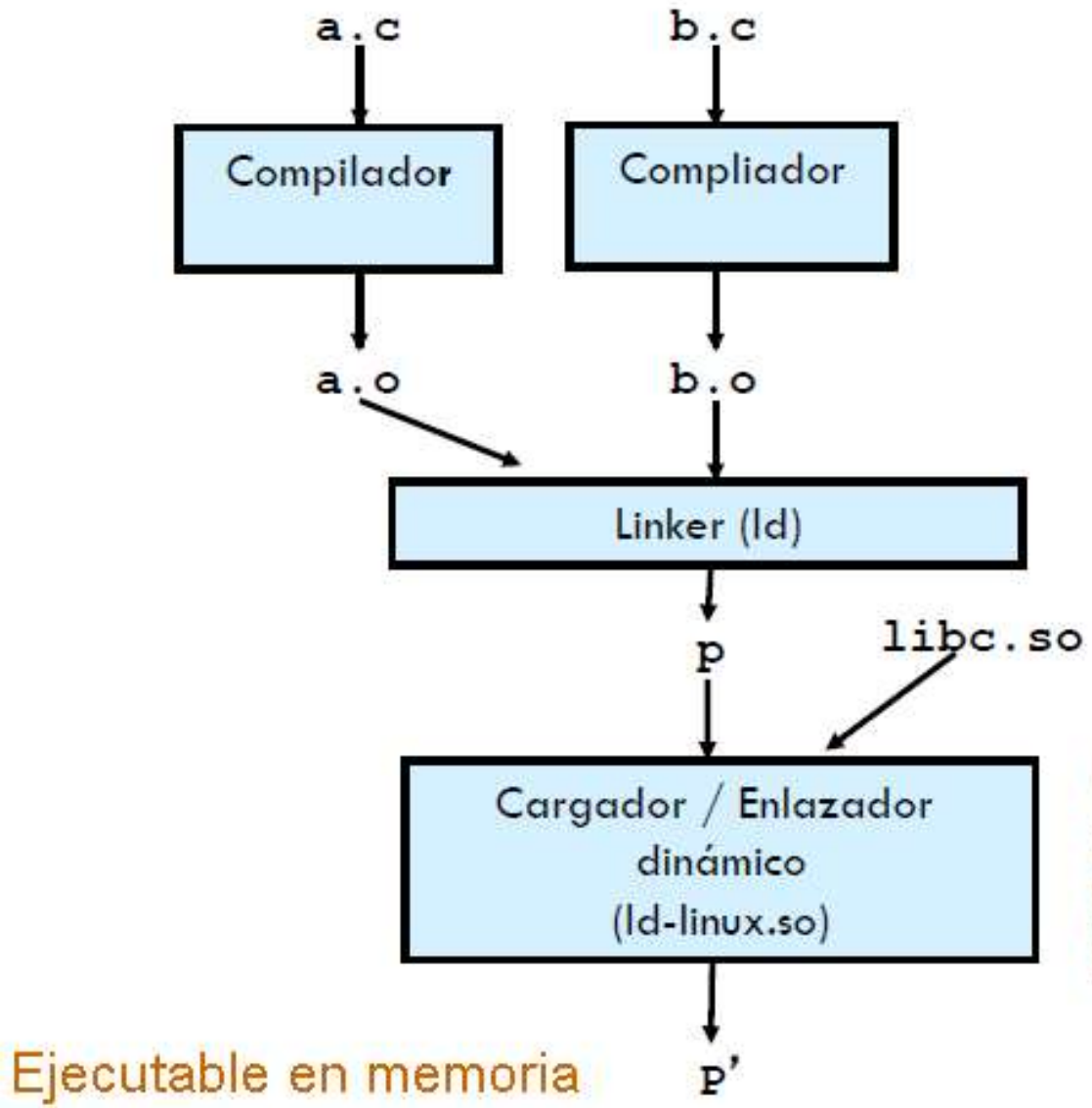
fichero objeto ejecutable: incluye el código de la funciones de libc que emplea

Bibliotecas estáticas y bibliotecas dinámicas

- Bibliotecas estáticas: desventajas:
  - código potencialmente duplicado en los ejecutables:
    - disco (sistema de ficheros)
    - espacio de memoria virtual de los procesos
  - bugs en las bibliotecas → nueva versión → re-enlazar

- Solución: **biliotecas dinámicas** (*.so) (dynamic link libraries, DLLs): componentes cargados en memoria y ejecutados en tiempo de ejecución:
  - las funciones de la librerías pueden ser compartidas por varios procesos

Bibliotecas dinámicas



Biblioteca compartida

las funciones de libc.so que invocan a.c y b.c se cargan, enlazan y están potencialmente compartidas entre procesos.

Concepto de Proceso

- Diferencia entre programa y proceso.
  - Un proceso es un programa en ejecución.
- El sistema operativo gestiona los procesos en ejecución (ciclo de vida de un proceso).
- Información del proceso constituida por: estado del procesador, imagen de memoria y BCP.
- La multitarea permite un mejor aprovechamiento de los recursos del computador.
- El cambio de contexto introduce una pequeña sobrecarga.
- Las bibliotecas estáticas se enlazan en tiempo de compilación y las dinámicas en tiempo de creación del proceso.
- La creación de un proceso implica la creación de su imagen de memoria y de su BCP.

Lecturas recomendada

- Básica
  - Carretero 2007:
    - 3.1 Concepto de proceso.
    - 3.2 Multitarea.
    - 3.3 Información del proceso.
    - 3.4 Vida de un proceso.
- Complementarias
  - Stallings 2005:
    - 3.1 ¿Qué es un proceso?
    - 3.3 Descripción de los procesos.
  - Siberschatz 2006:
    - 3.1 Concepto de proceso.
    - 3.3 Operaciones sobre los procesos.