

# SISTEMAS OPERATIVOS: PROCESOS

Introducción a la gestión de Procesos

# ADVERTENCIA

2

- Este material es un simple gui3n de la clase: no son los apuntes de la asignatura.
- El conocimiento exclusivo de este material no garantiza que el alumno pueda alcanzar los objetivos de la asignatura.
- Se recomienda que el alumno utilice los materiales complementarios propuestos.

# Contenido

3

- **Concepto de proceso.**
- Ciclo de vida básico de un proceso.
- Información de un proceso.
- Multitarea.
- Cambio de contexto.
- Generación de ejecutables.

# Concepto de proceso

4

## □ Proceso: Programa en ejecución.

- Cada ejecución de un programa da lugar a un proceso.
- El proceso → unidad de procesamiento que gestiona el sistema operativo.

No es un ejecutable, pero el proceso ejecutado tiene los valores de entorno del ejecutable.

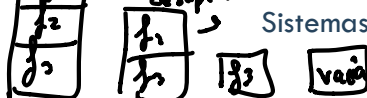


## □ Un proceso está formado por:

- Código del programa: Instrucciones. text

- Conjunto de datos asociados a la ejecución del programa

En pila se van guardando los llamados y cuando acaba se desapila. Si no desapila se llena y no caberá más, y da stack overflow.



Sistemas Operativos – Introducción a Procesos

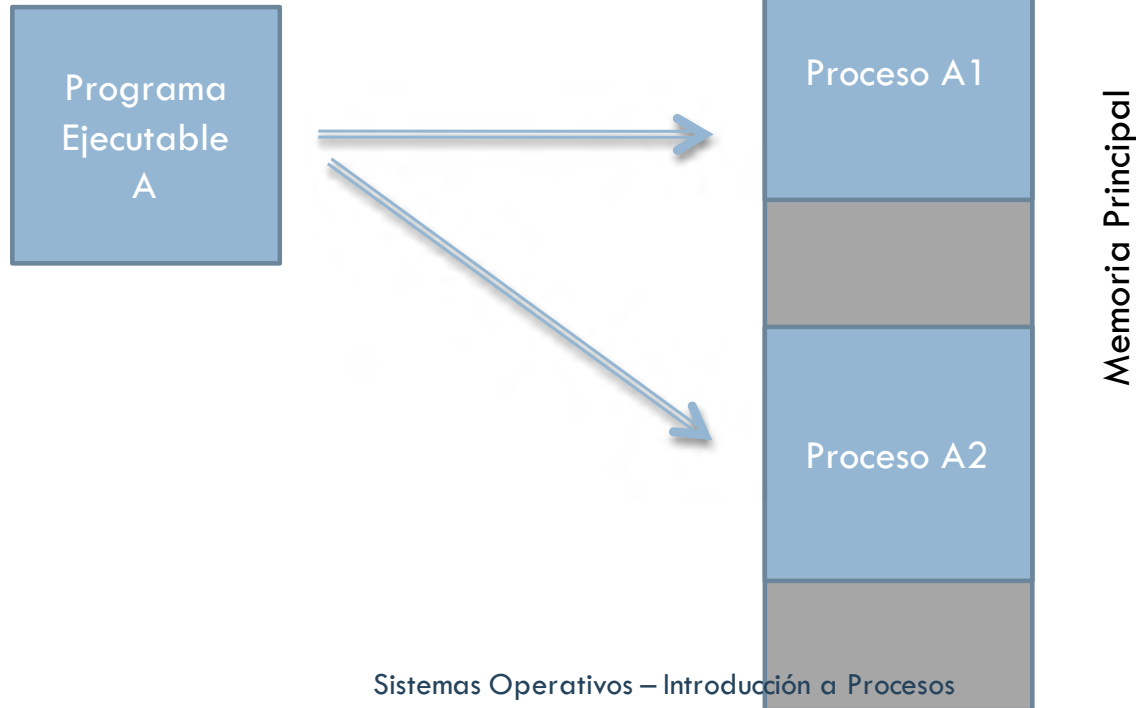
memoria dinamica  
heap  
memoria estatica  
static

Se asigna con malloc (1; size);  
free (obj);  
un segmento de memoria y se libera con

# Ejecución de programas

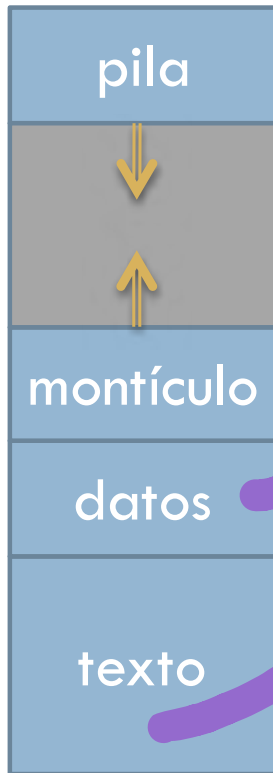
5

Cada proceso tiene un identificador llamado pid (process id), que nos permite obtener más información sobre el proceso, el contexto (directorio, padre, ...)



# Representación en memoria

6



- Un proceso necesita memoria para las instrucciones y los datos.
- Distintas instancias de un programa necesitan zonas independientes para los datos.

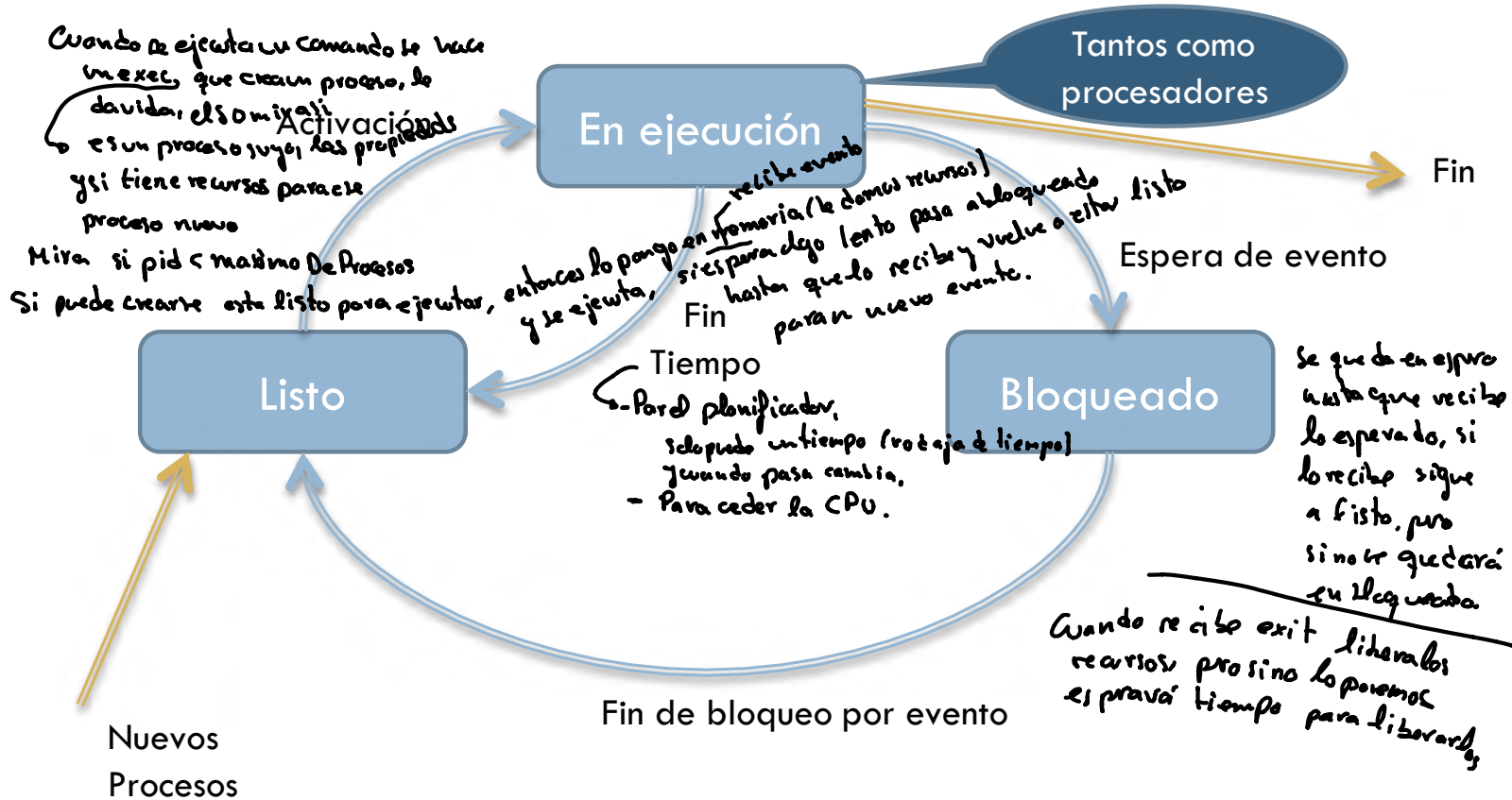
# Contenido

7

- Concepto de proceso.
- **Ciclo de vida básico de un proceso.**
- Información de un proceso.
- Multitarea.
- Cambio de contexto.
- Generación de ejecutables.

# Ciclo de vida básico de un proceso

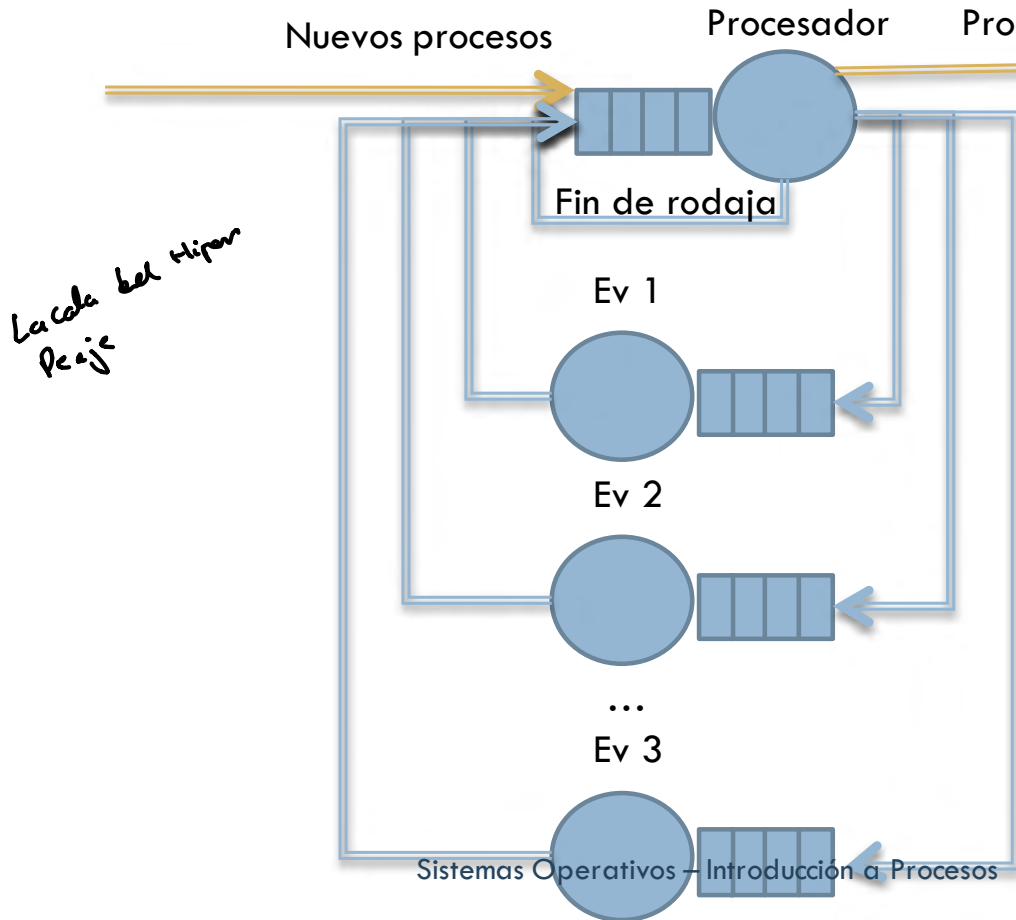
8





# Modelo de colas simplificado: <sup>FIFO</sup> Un procesador

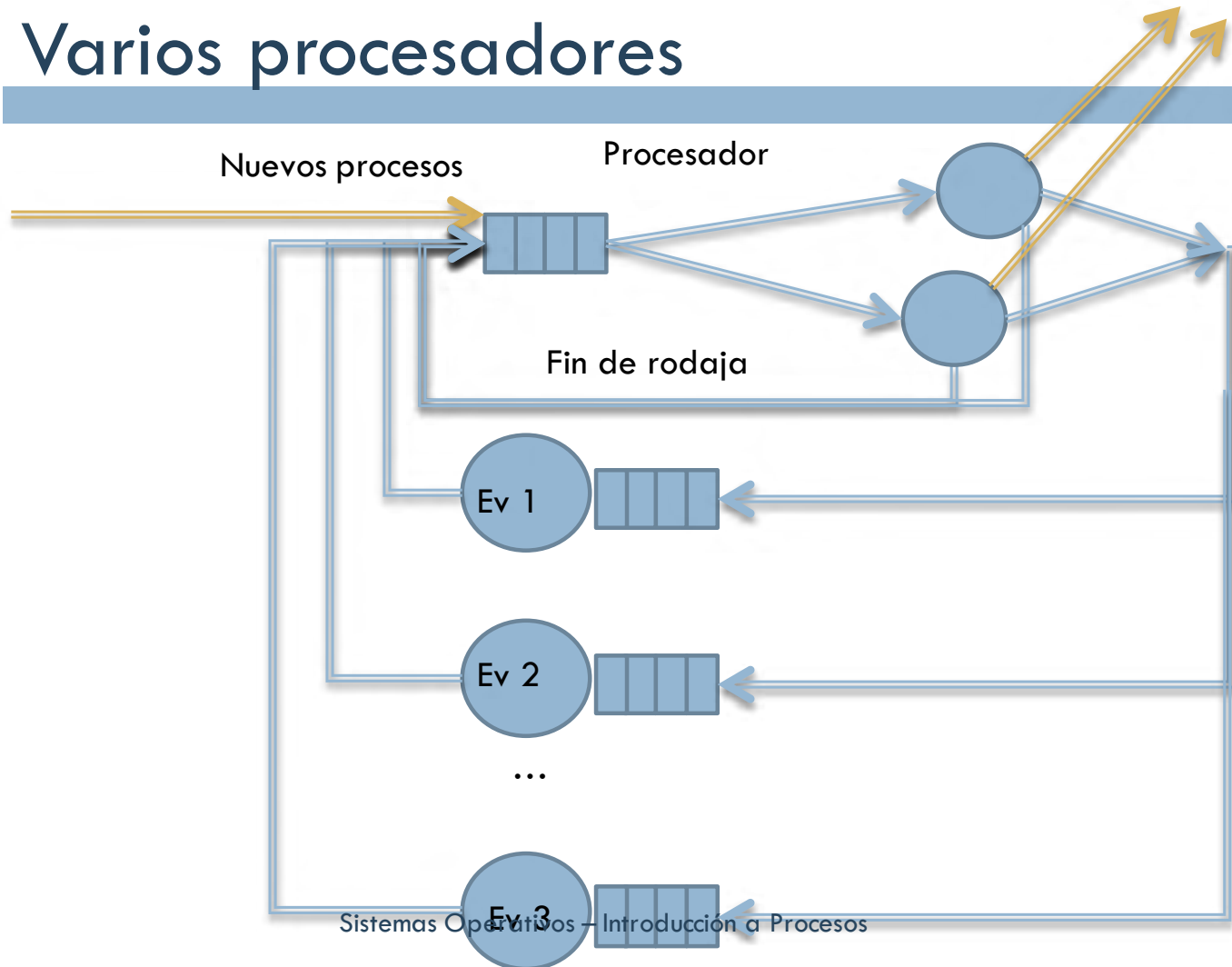
9



Estos en una cola, y según los unidades de procesado va a uno u otro, hacen de procesos más rápidos y otros para más pesados. El que elige donde van unos u otros es el dispatcher.

# Modelo de colas simplificado: Varios procesadores

Procesos terminados



# Contenido

11

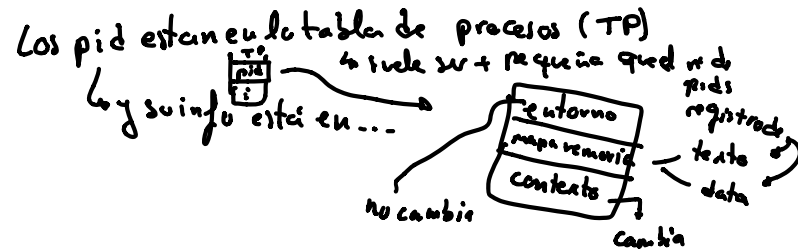
- Concepto de proceso.
- Ciclo de vida básico de un proceso.
- **Información de un proceso.**
- Multitarea.
- Cambio de contexto.
- Generación de ejecutables.

# Información del proceso

12

init pid=1  
internet daemon pid=2  
impresora daemon pid=3  
+1  
+1  
Cada uno tiene un número, no se reutilizan. Habría que resetear si aunque haya muerto.

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

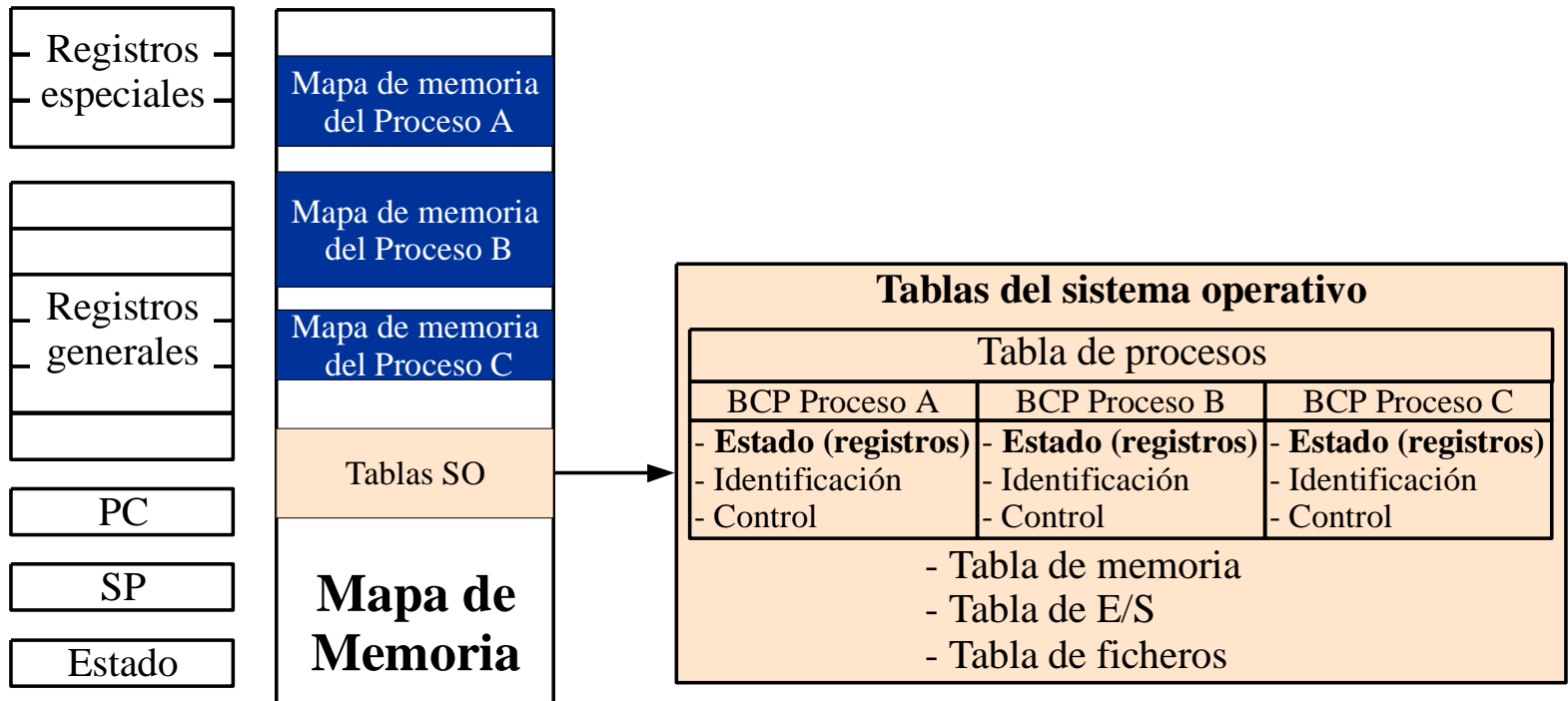


- Tres categorías:
  - Información almacenada en el procesador.
  - Información almacenada en memoria.
  - Información adicional gestionada por el sistema operativo.

Hay que salvar el contexto antes de bloquear o de salir a otro proceso.  
Cuando vuelva a este proceso restaura el contexto.

# Información del proceso

13



# Estado del procesador

14

- 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

15

- 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~~

San leitos  
swap/—

Memoria virtual: Engaña, ya que necesita recursos que no tiene la máquina, y tiene que matar otro para ir haciendo huecos (busca víctimas)

16

- 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.

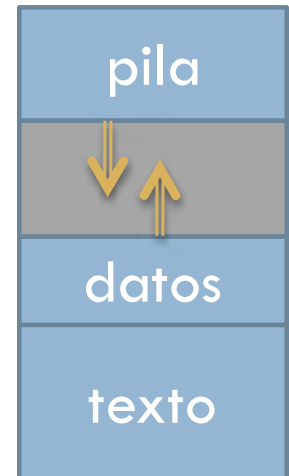


# Modelos de imagen de memoria:

## Regiones múltiples

17

- 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.

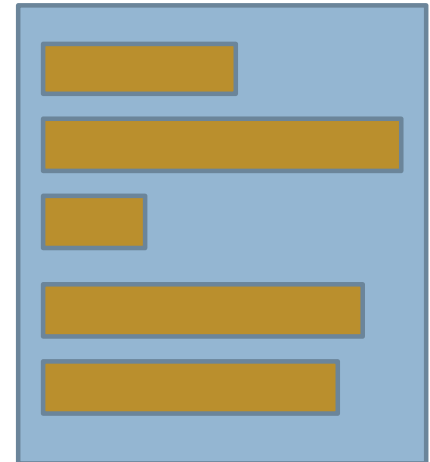


# Modelos de imagen de memoria:

## Regiones múltiples

18

- 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

19

- 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

20

- 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.

# Información fuera del BCP

21

- 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.

# Tabla de páginas *Mirany*

22

- 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

23

- Se sitúan fuera del BCP.

Quando se abre un fichero, en el proceso  
hay una <sup>como entrada en la</sup>  
Esta tabla es propia de cada  
proceso, excepto padre e hijo.

- Si se añaden a la tabla de ficheros abiertos (en el BCP) no se pueden compartir. <sup>lo donde se indican los abiertos.</sup>  
<sup>El descriptor es la posición del fichero en esa tabla.</sup>
- Si se asocian al nodo-i se comparten siempre. <sup>la tabla</sup>
- Se ponen en una estructura común a los procesos y se asigna uno nuevo en cada servicio OPEN.

# Ejemplo: Ejecución de un mandato

24

```
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char** argv) {
    pid_t pid;
    pid = fork();
    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;
}
```

prog cat fl

*El padre puede modificar el entorno que tendrá el hijo.*

*→ El hijo tiene que decirle al padre que ha terminado, pero como el padre no espera, sino que se va con return, el hijo que da cambio, para evitarlo el padre lo cambiamos a que sea init.*



# Servicio fork

*Comparten contexto*

25

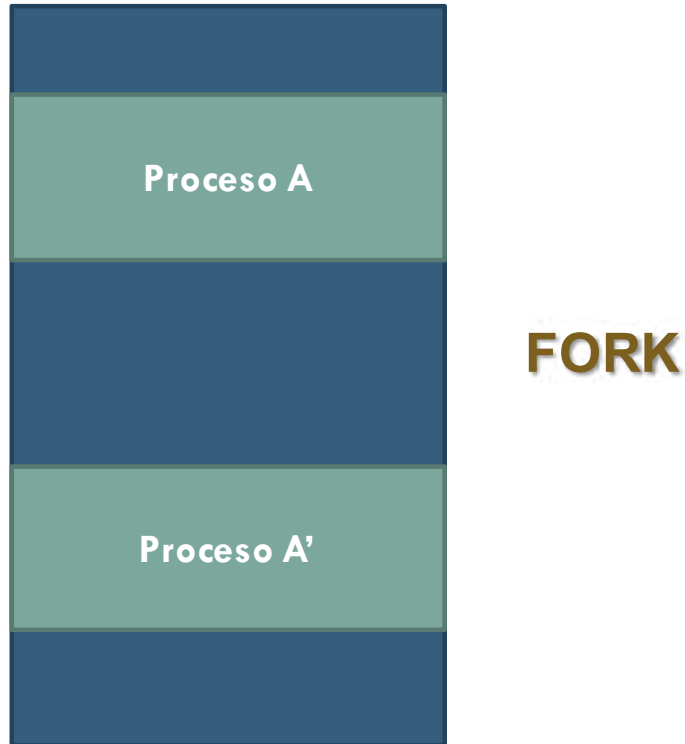
- `pid_t fork(void);`

*Solo copiará lo del padre cuando lo modifique, pero si solo lee, comparte espacio con el padre*

- 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

# Servicio fork

26



# Servicio exec

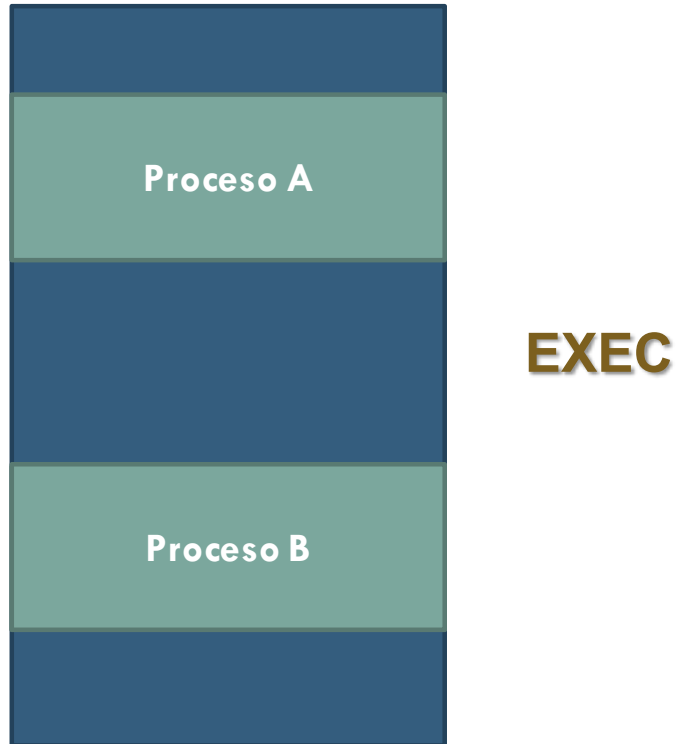
Lee un ejecutable, pero cambia el mapa de memoria y el contexto al del ejecutable.

27

- Servicio único pero múltiples funciones de biblioteca.
- ```
int exec1(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

28



# Servicio exit

29

- 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.

# Contenido

30

- Concepto de proceso.
- Ciclo de vida básico de un proceso.
- Información de un proceso.
- **Multitarea.**
- Cambio de contexto.
- Generación de ejecutables.

# Tipos de sistemas operativos

31

## Sistemas Operativos

Multiproceso  
(varios procesos en ejecución)

*Paralelismo*

Monoproceso  
(un único proceso)

*Concurrencia*

Multiusuario  
(varios usuarios  
a la vez)

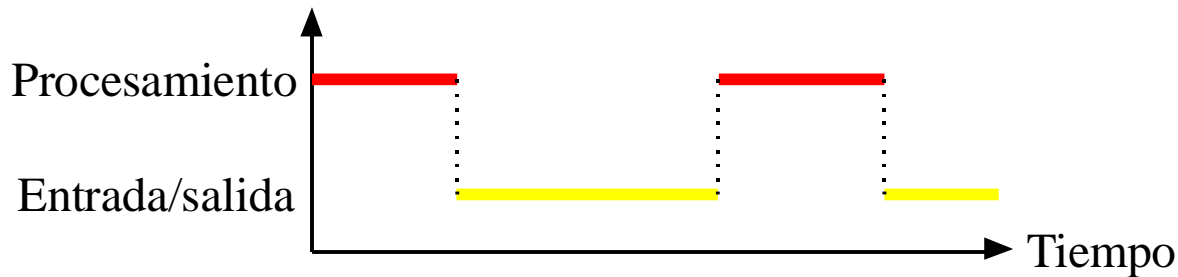
Monousuario  
(un único  
usuario a la  
vez)

Monousuario  
(un único usuario a la  
vez)

# Principios de la multitarea

32

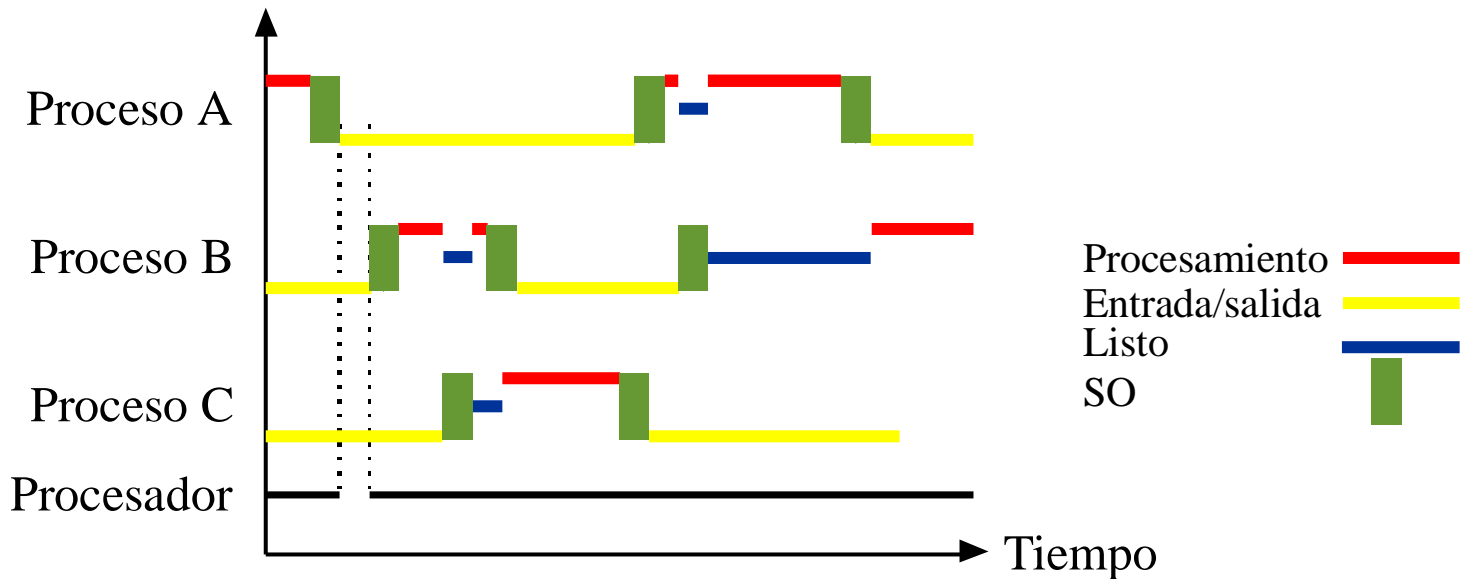
- **Paralelismo real** entre E/S y UCP (DMA)
- Alternancia en los procesos de fases de E/S y de procesamiento
- La memoria almacena varios procesos





# Ejecución en un sistema multitarea

33



# Ventajas de la multitarea

34

- 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

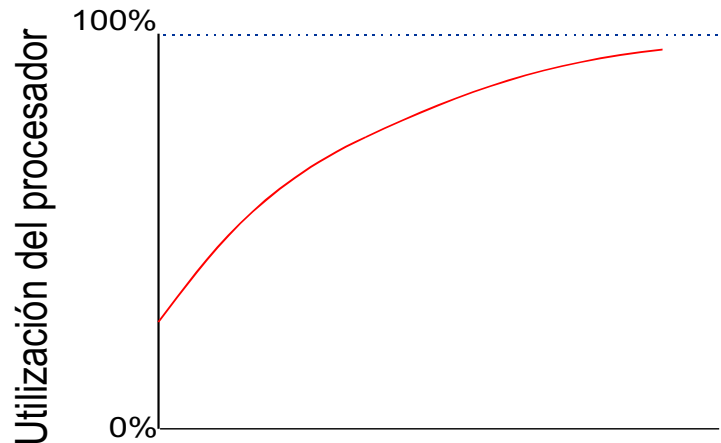
35

- Grado de multiprogramación: n° de procesos activos  
*El rendimiento de la máquina disminuye*
- Necesidades de memoria principal: Sistema sin memoria virtual

|           |
|-----------|
| Proceso A |
| Proceso B |
| Proceso C |
| SO        |

Memoria principal

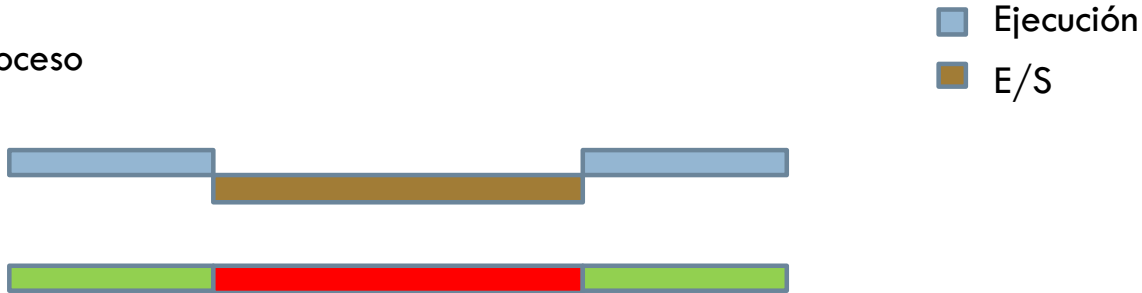
Cada proceso reside totalmente en M.p



# Multiprogramación: uso de la CPU

36

1 proceso



2 procesos



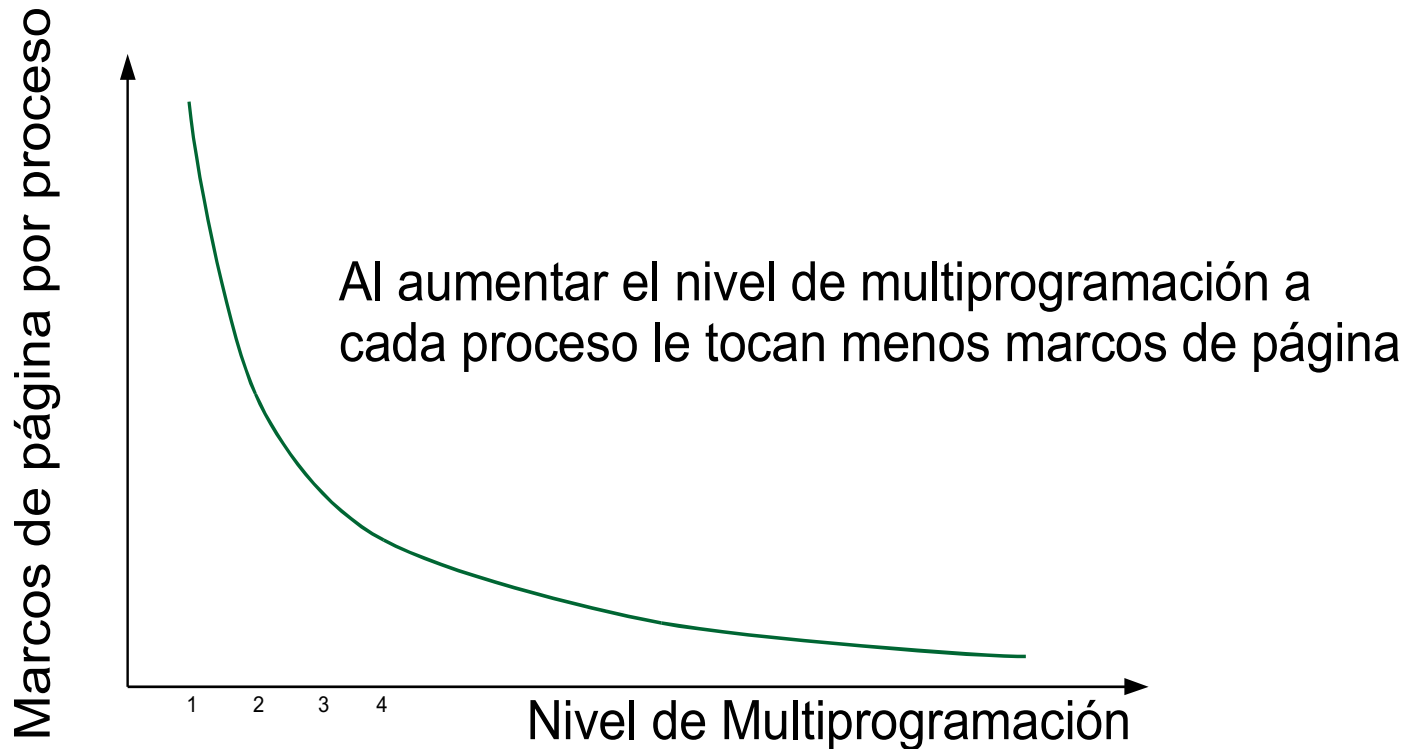
# Multiprogramación y memoria virtual

37

- 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

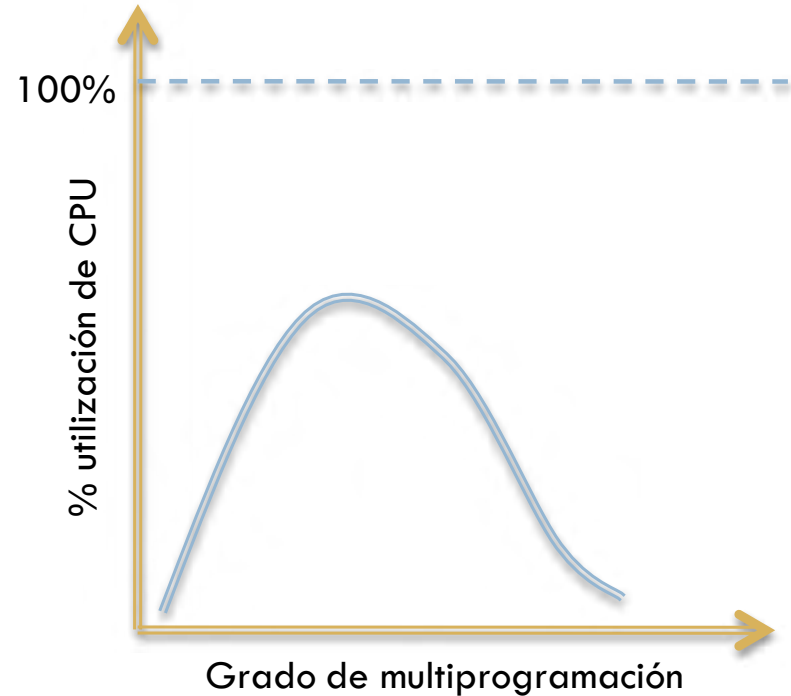
38



# Rendimiento: Poca memoria física

39

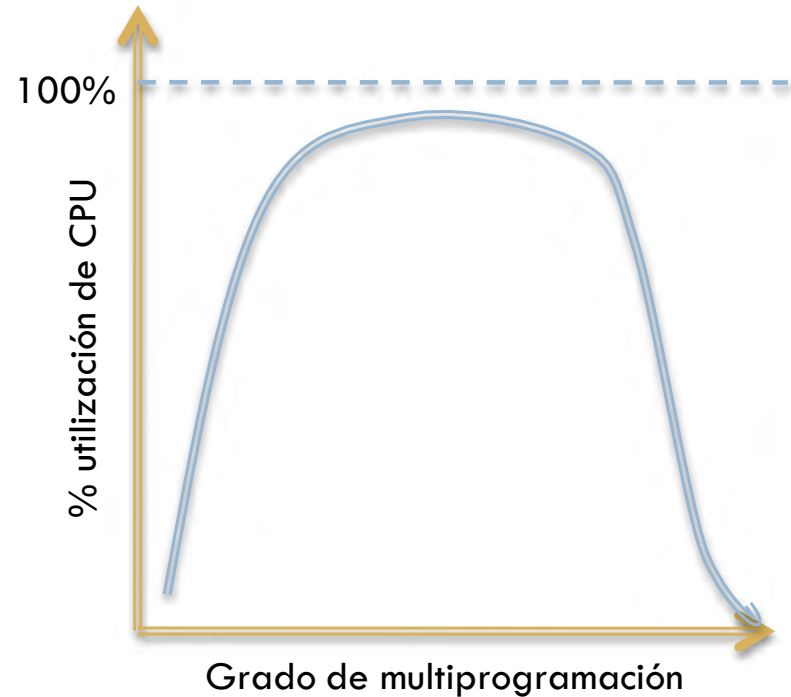
- Al aumentar el grado de multiprogramación:
  - Desciende el tamaño del conjunto residente de cada proceso.
- Se produce hiperpaginación antes de alcanzar un porcentaje alto de uso de CPU.
- **Solución:** Ampliación de memoria principal.



# Rendimiento: Mucha memoria física

40

- 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.





# Contenido

41

- Concepto de proceso.
- Ciclo de vida básico de un proceso.
- Información de un proceso.
- Multitarea.
- **Cambio de contexto.**
- Generación de ejecutables.

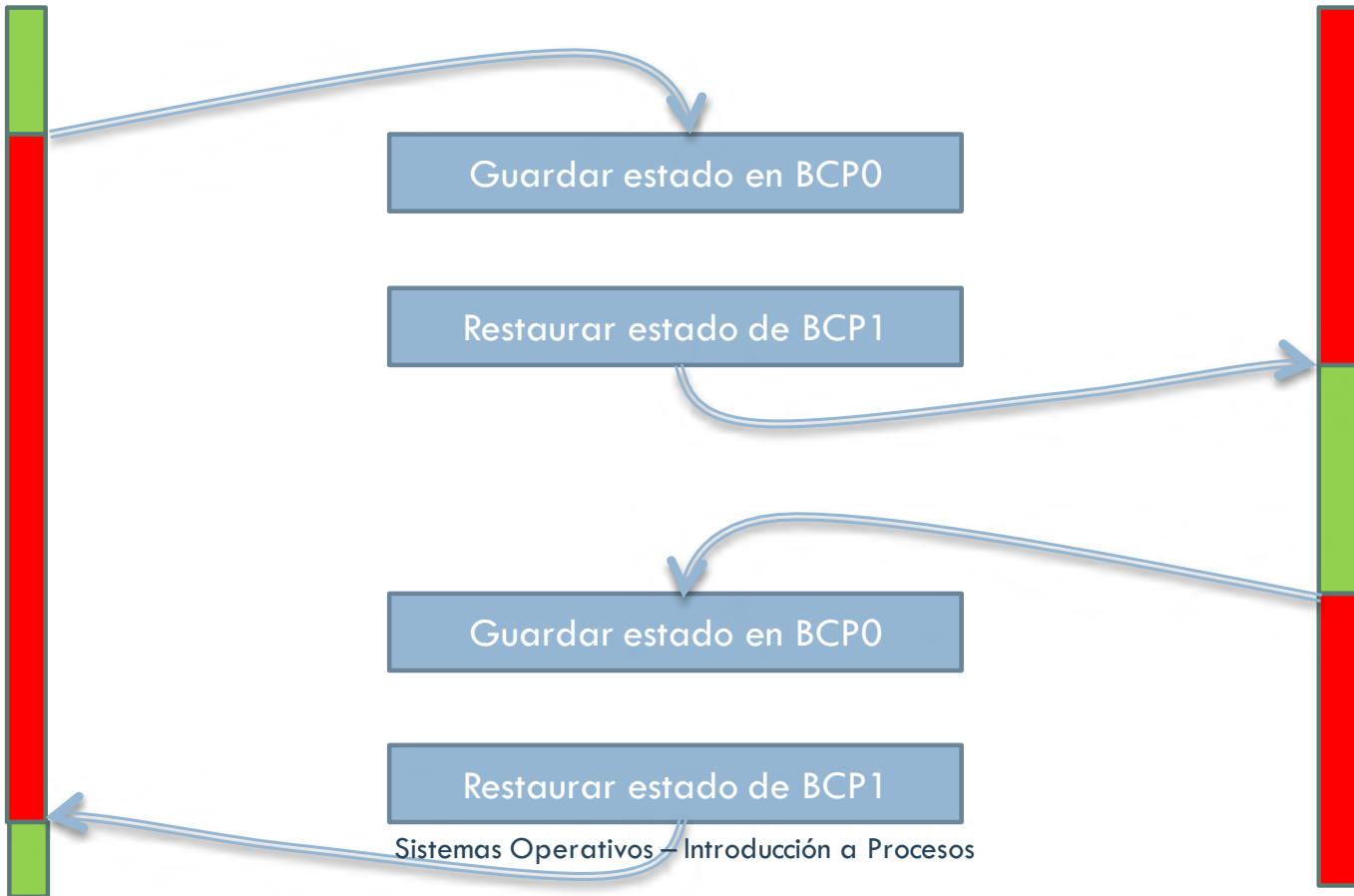
# Cambios de contexto

42

- 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

43



# Tipos de cambio de contexto

44

## □ 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*

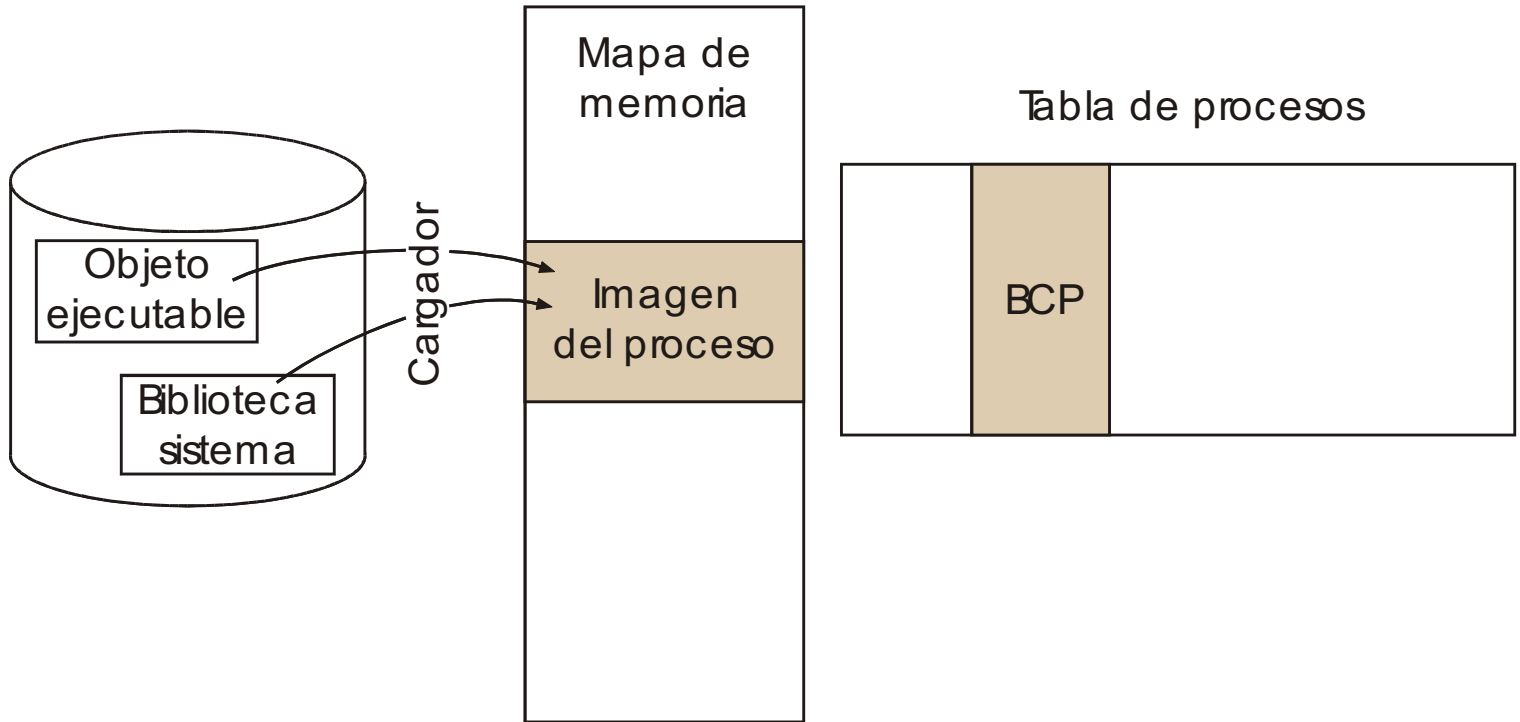
# Contenido

45

- ❑ Concepto de proceso.
- ❑ Ciclo de vida básico de un proceso.
- ❑ Información de un proceso.
- ❑ Multitarea.
- ❑ Cambio de contexto.
- ❑ **Generación de ejecutables.**  Mi rol en casa

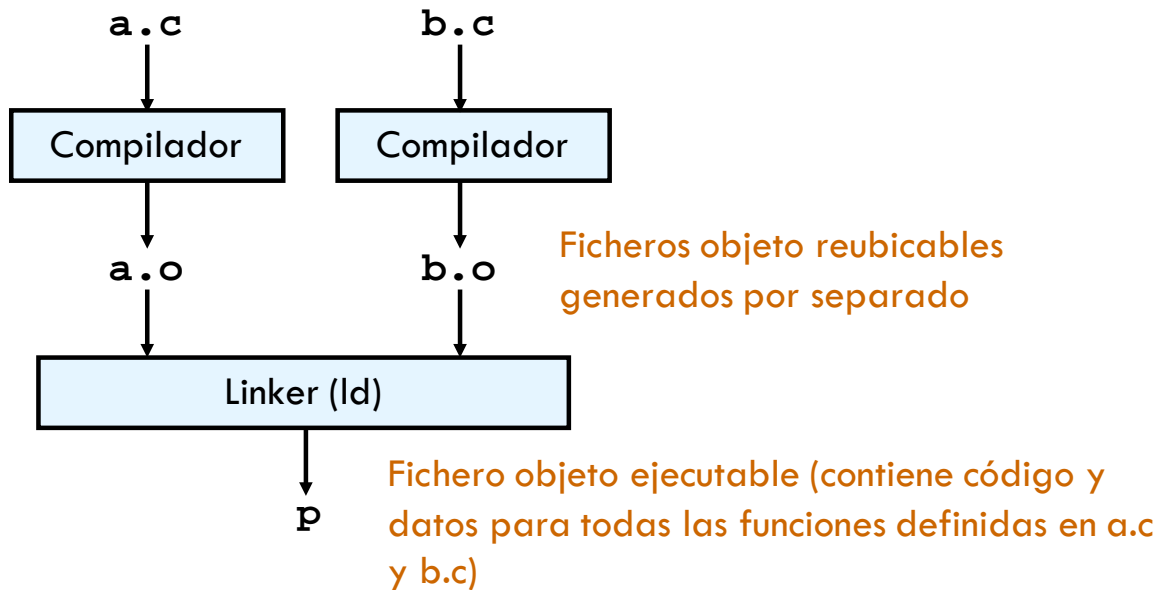
# Formación de un proceso

46



# Generación de ejecutables

47



# Editor de enlaces (*linker*)

48

- 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

49

- 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

50

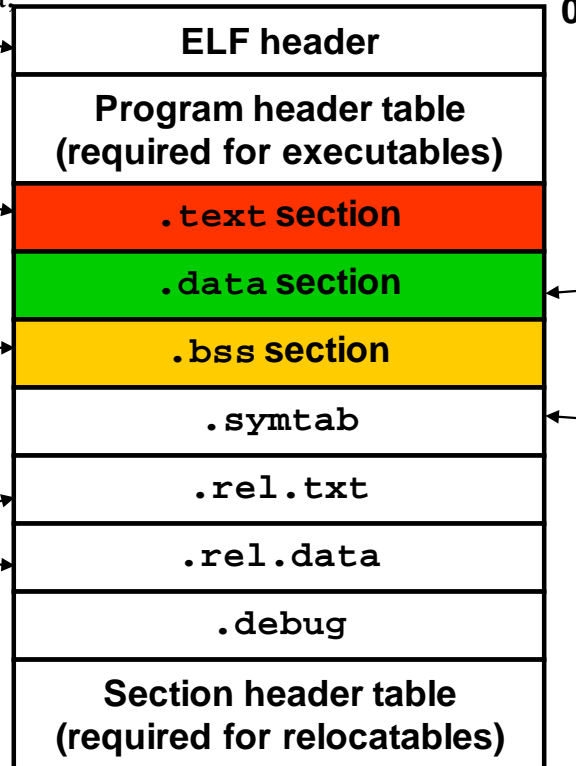
Tipo (.o, .exe, .so, etc); máquina:

Big /Little end; etc

Código

“block started by  
symbol” Datos  
estáticos no  
inicializados

Info. de  
reubicación

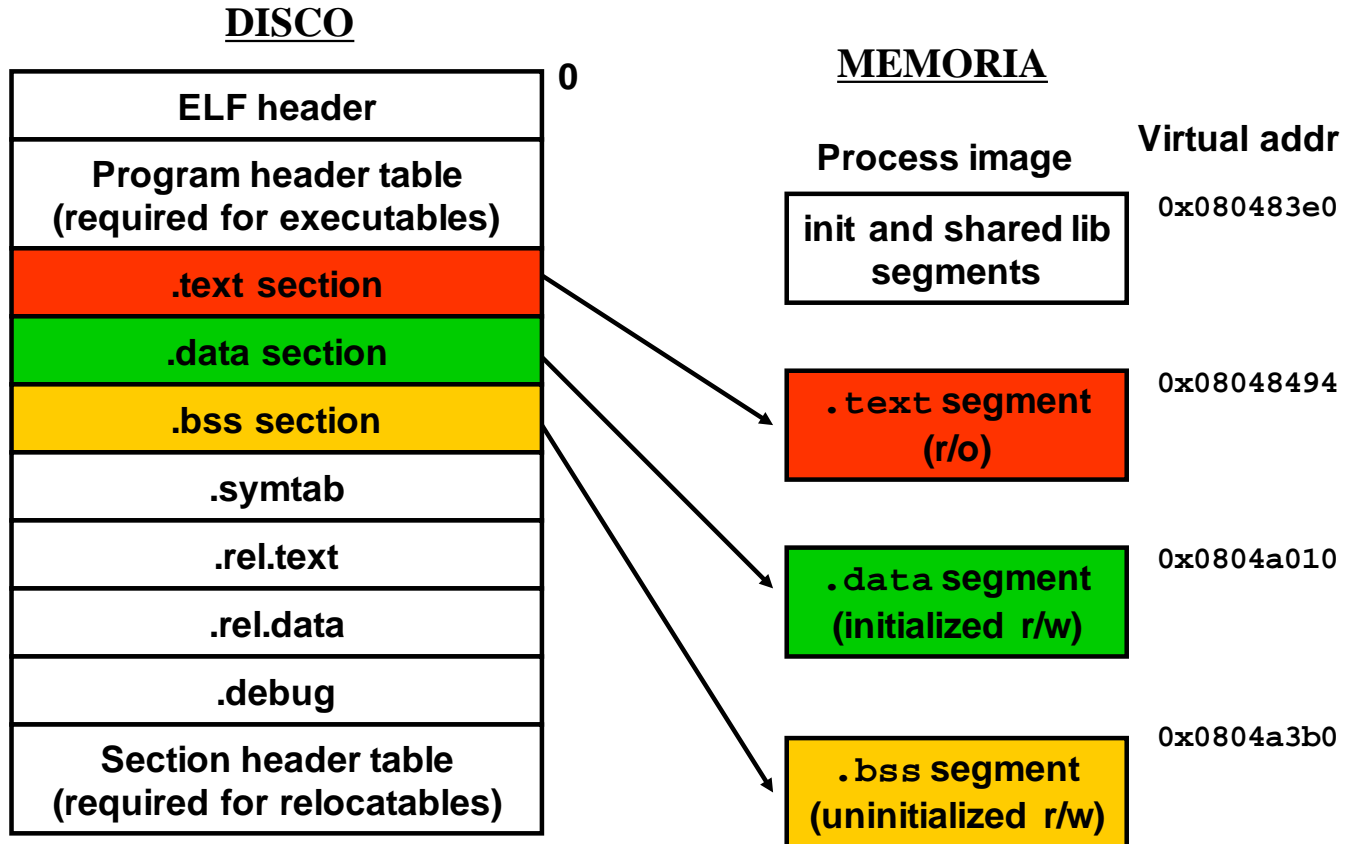


Datos estáticos  
inicializados

Tabla de  
símbolos

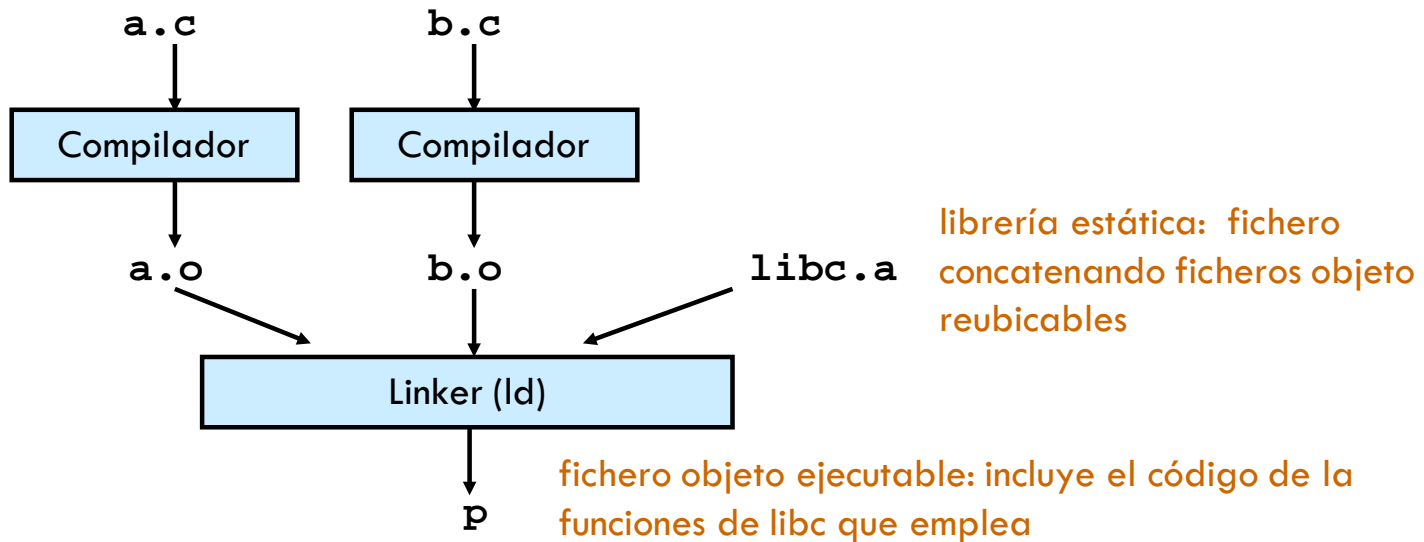
# Carga de ejecutable

51



# Bibliotecas estáticas

52



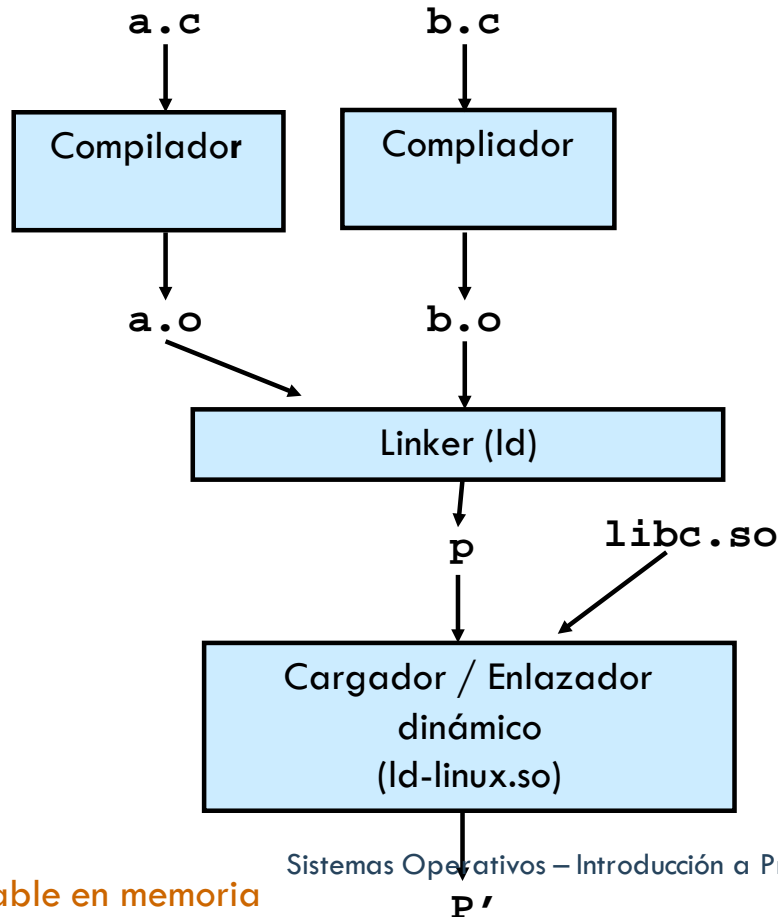
# Bibliotecas estáticas y bibliotecas dinámicas

53

- 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: **bibliotecas 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

54



Biblioteca compartida

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

# Puntos a recordar

55

- 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

56

## 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.