SISTEMAS OPERATIVOS: PROCESOS

Introducción a la gestión de Procesos

Concepto de proceso

- Proceso: Programa 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

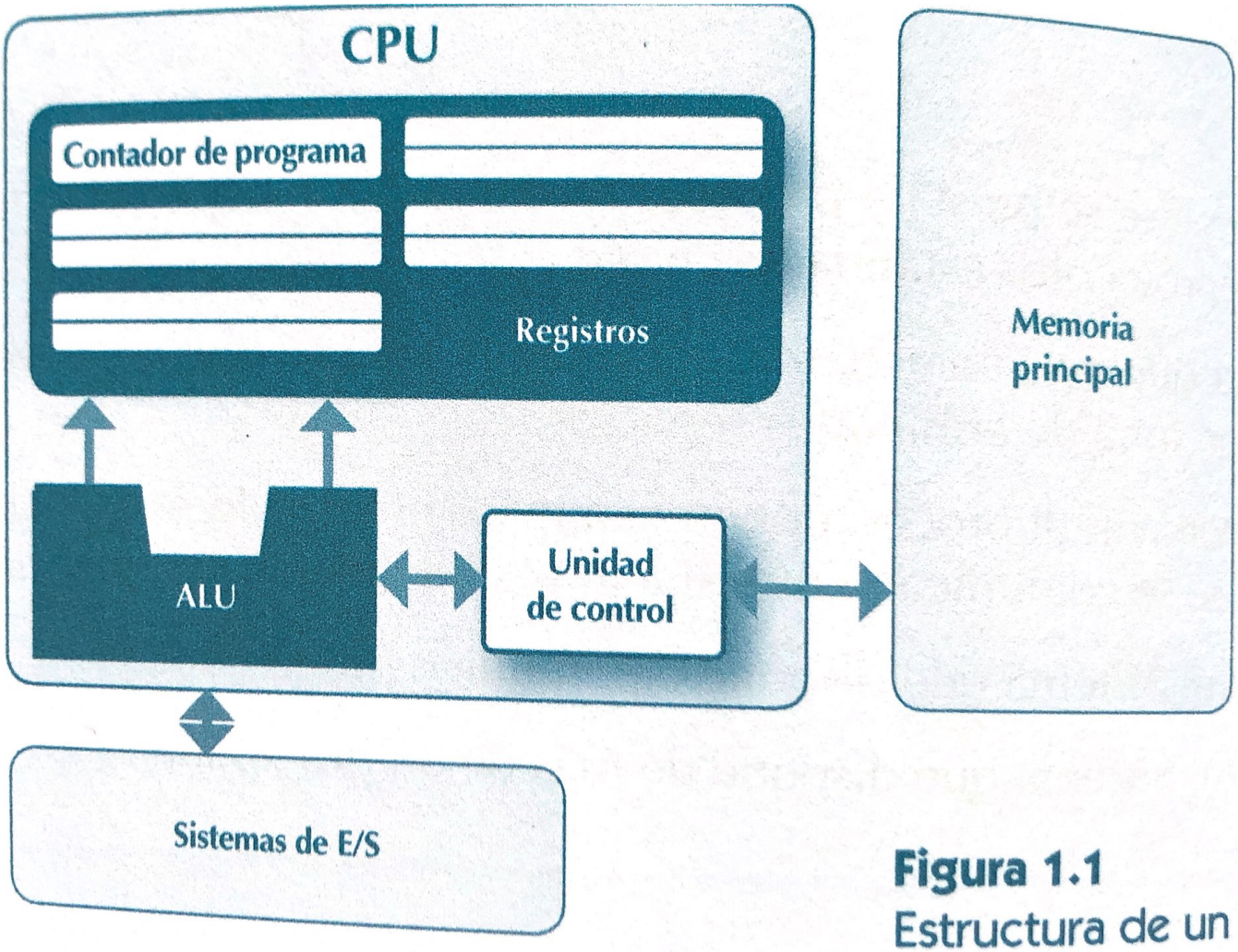
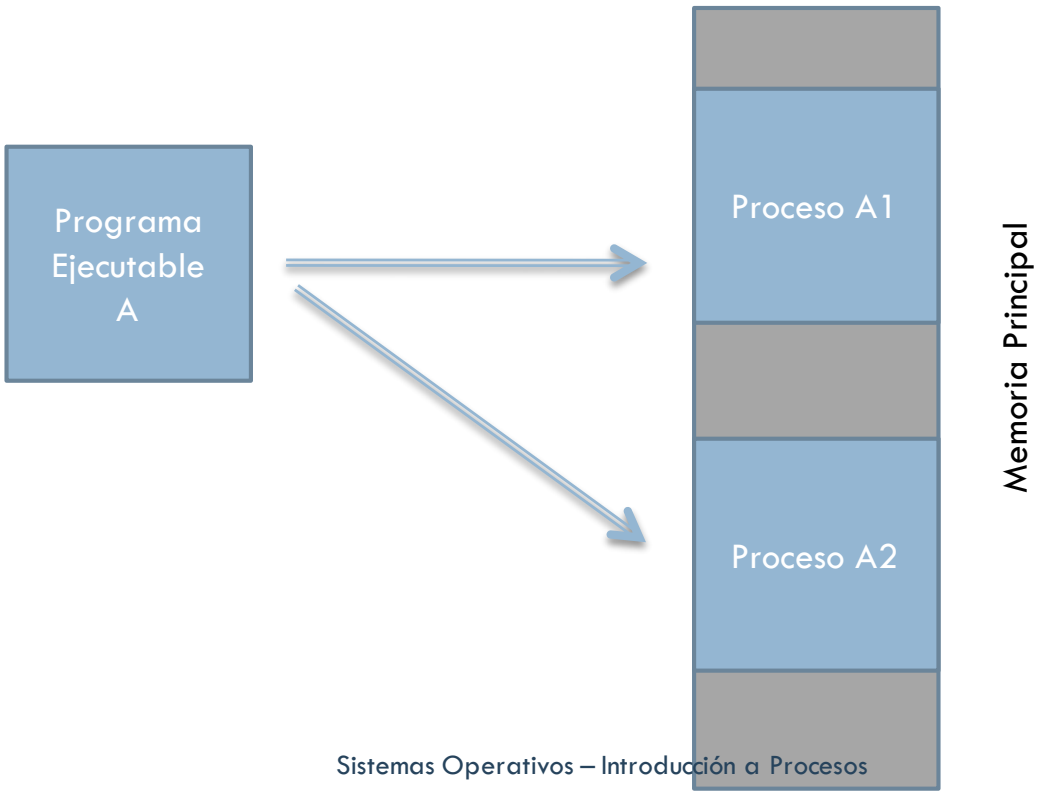
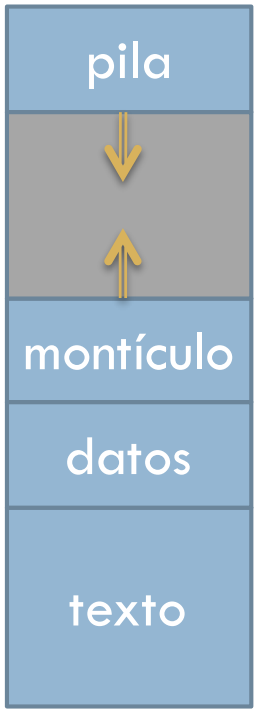


Figura 1.1
Estructura de un proces

Ejecución de programas

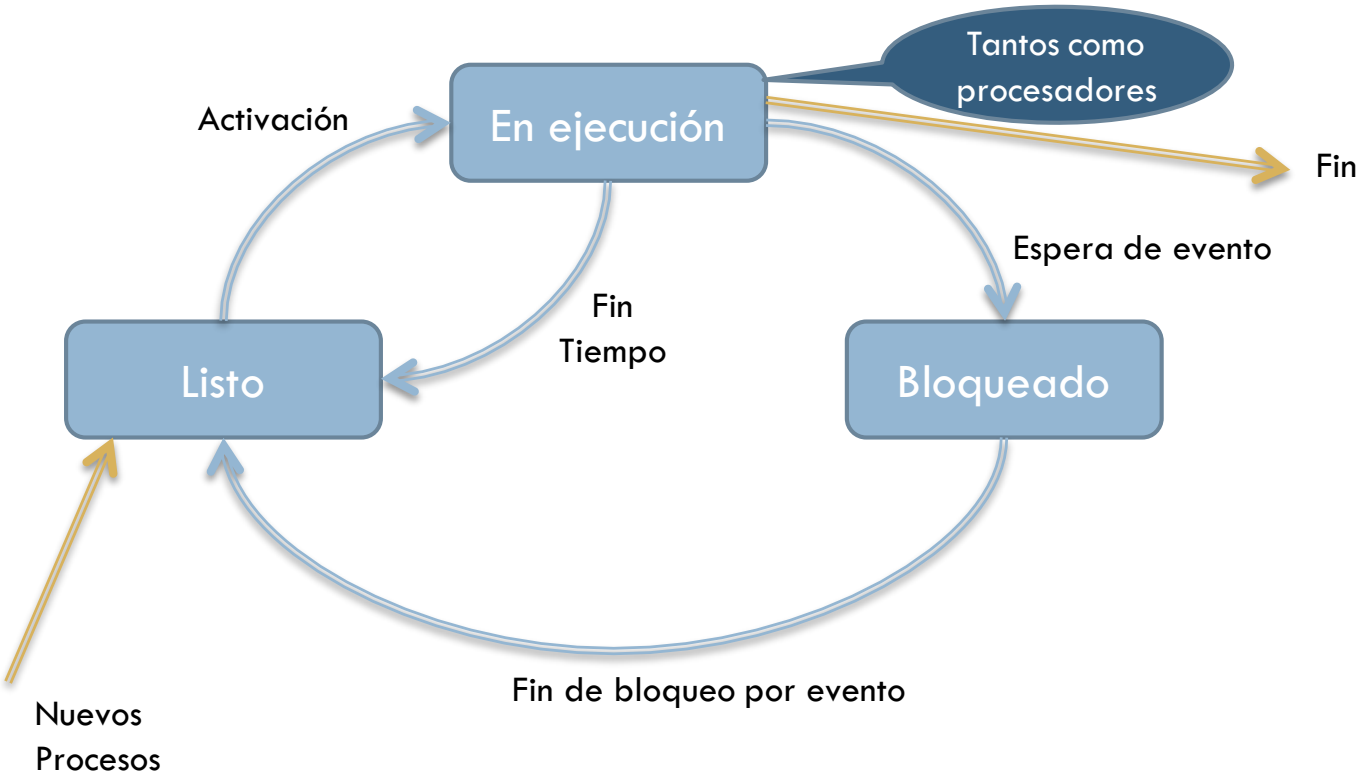


Representación en memoria

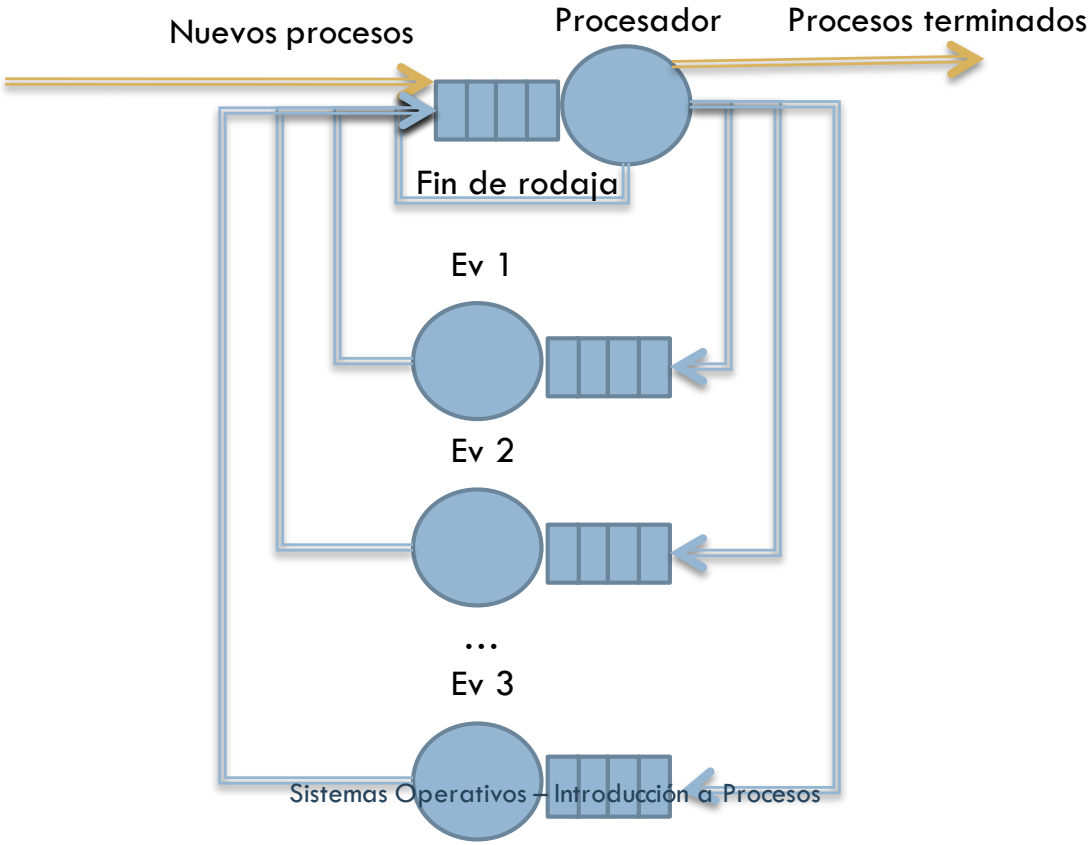


- Un proceso necesita memoria para las instrucciones y los datos.
- Distintas instancias de un programa necesitan zonas independientes para los datos.

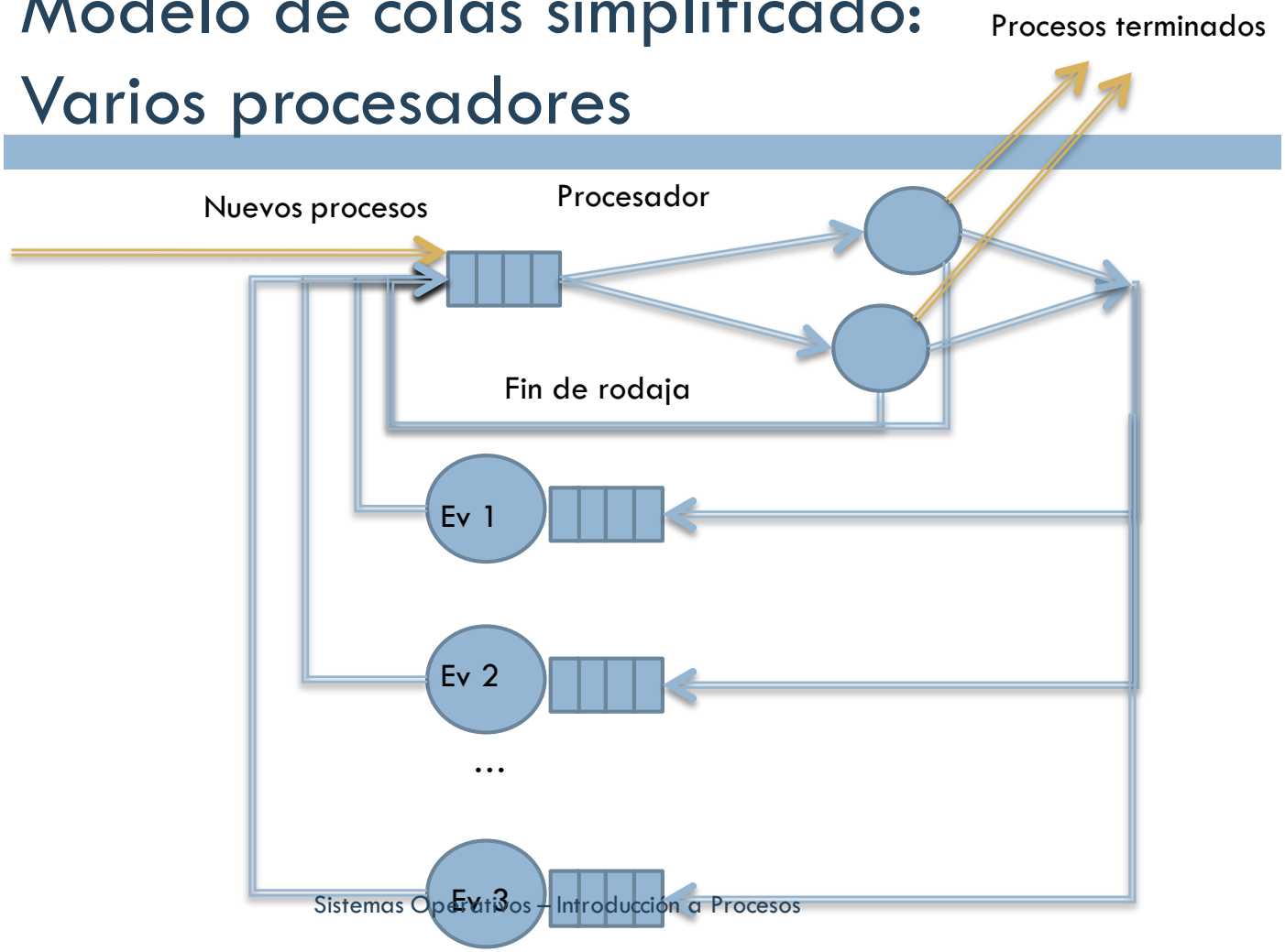
Ciclo de vida básico de un proceso



Modelo de colas simplificado: Un procesador



Modelo de colas simplificado: Varios procesadores

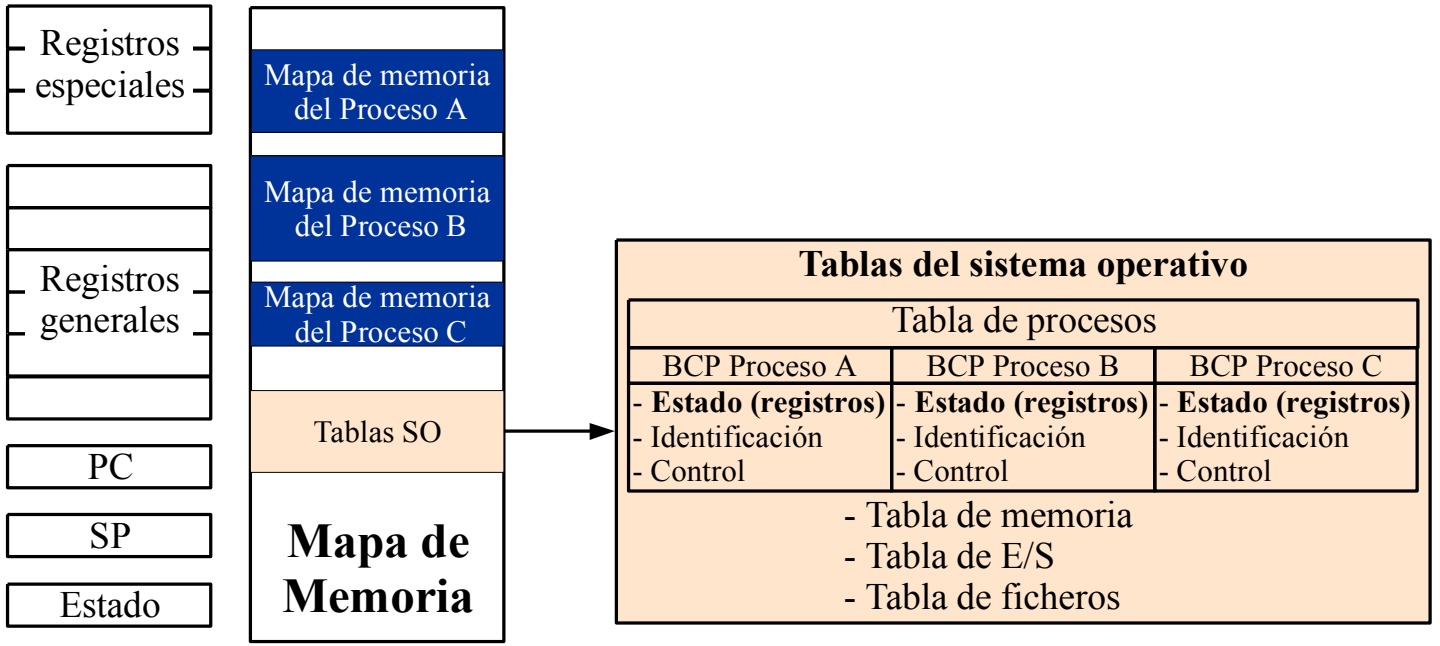


Información del proceso

- 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.

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.

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

Descripción de regiones asigrnada.

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

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

Información para comunicación entre procesos.

Ejemplo: Ejecución de un mandato


```
#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 f1

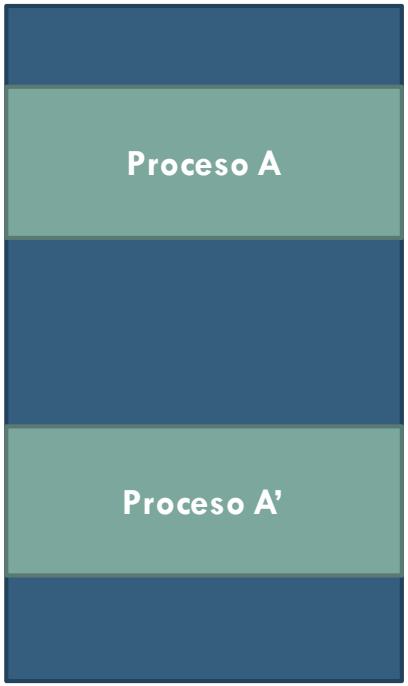
Servicio fork

- `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

Servicio fork



FORK

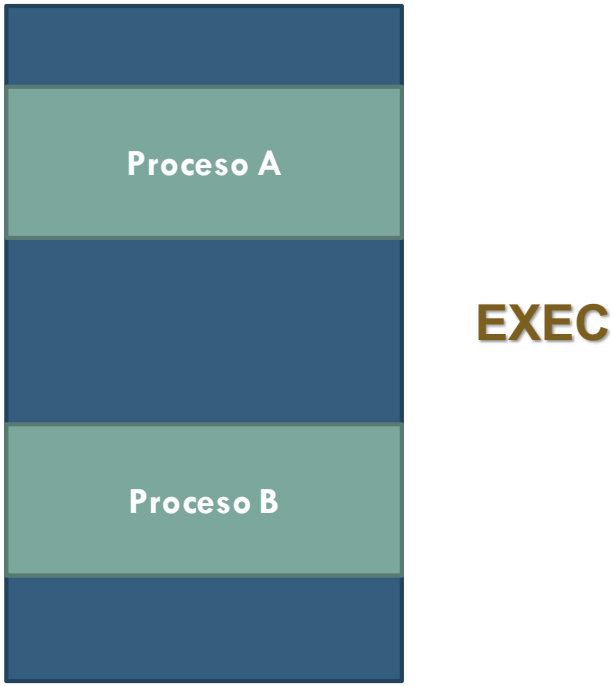
Servicio exec

- 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

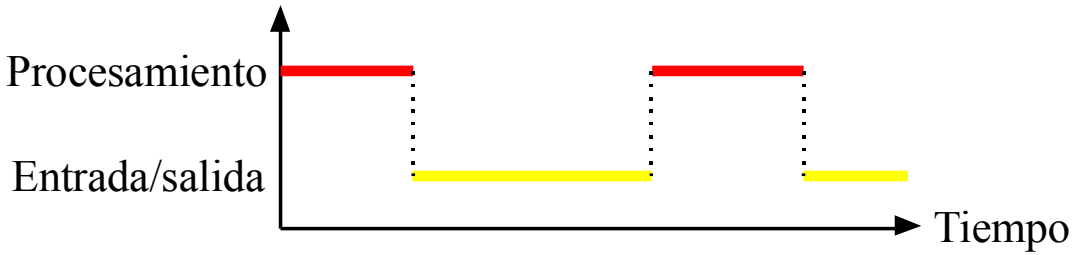
- Finaliza la ejecución del proceso.

- `void exit(status);`

- Se cierran todos los descriptores de ficheros abiertos.
- Se liberan todos los recursos del proceso.
- Se libera el BCP del proceso.

Principios de la multitarea

- 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



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.

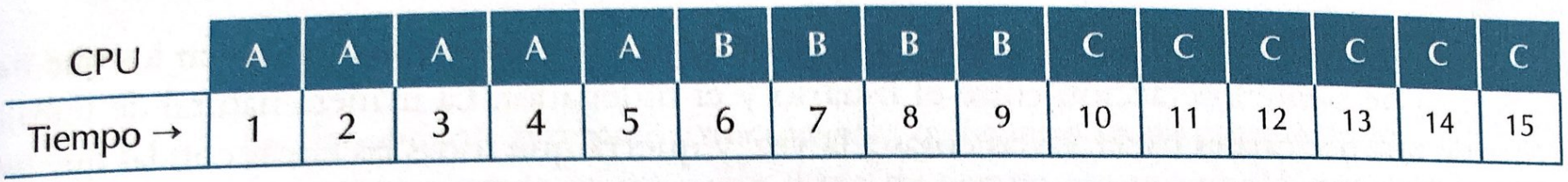


Figura 1.2
Ejecución de procesos sin multitarea.

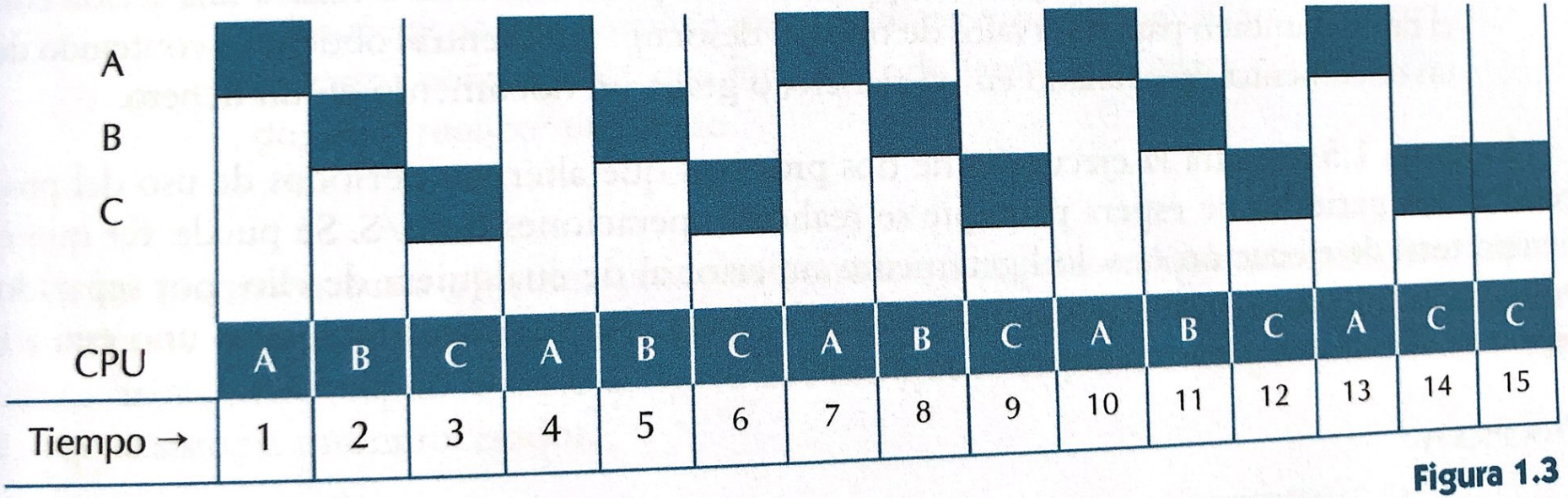
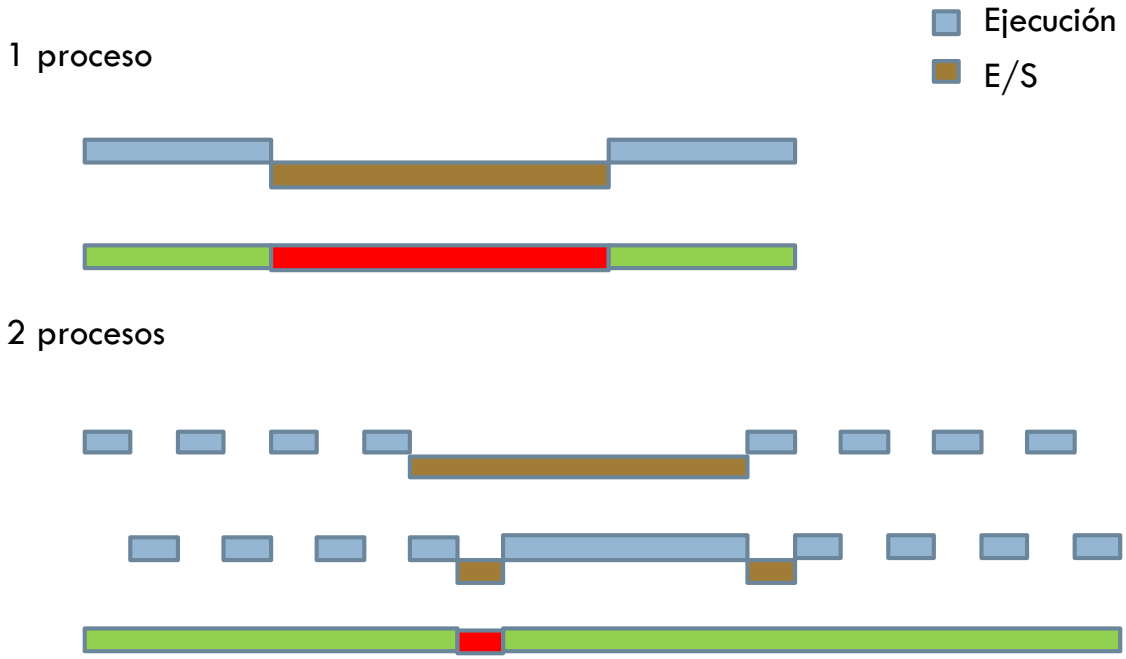


Figura 1.3
Ejecución de procesos con multitarea.

Multiprogramación: uso de la CPU

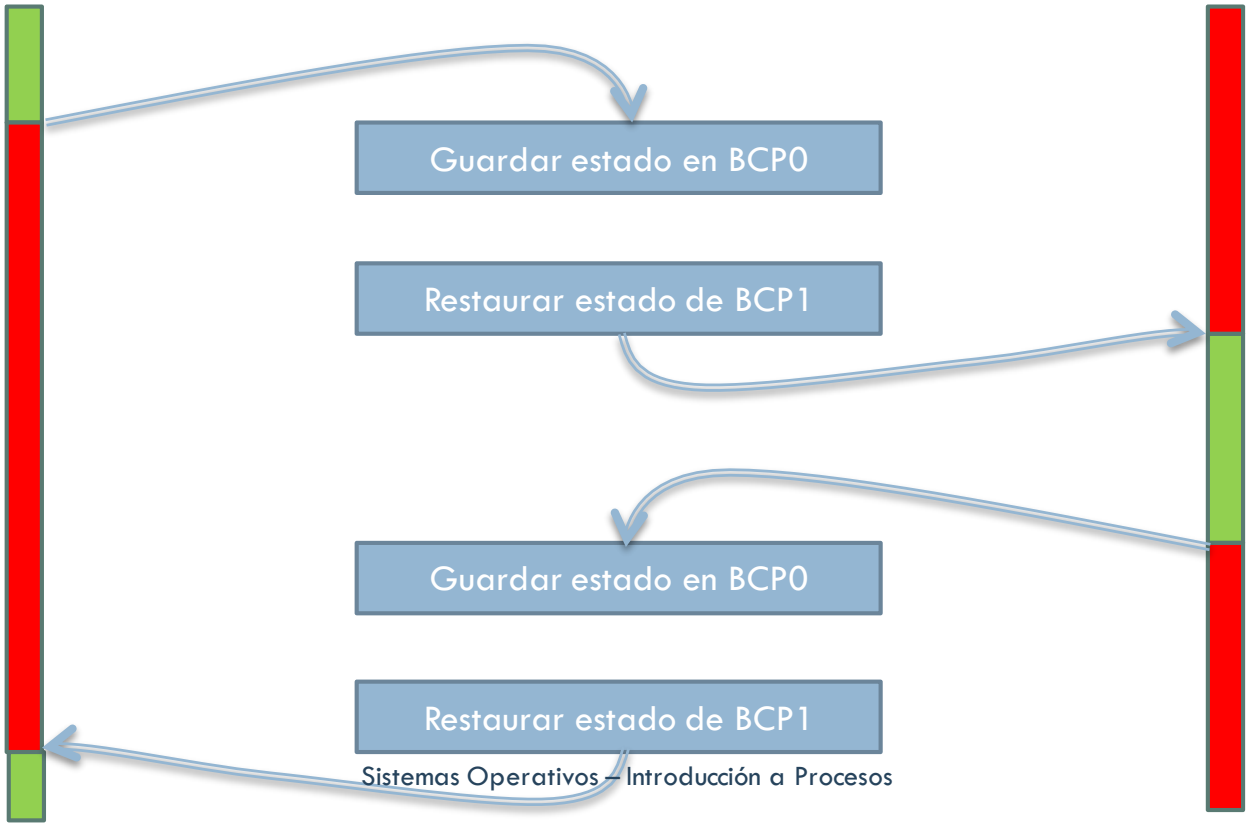


Cambios de contexto

- 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*