

# Lección 3

## Procesos e hilos

Sistemas Operativos  
Ingeniería Informática

# Lecturas recomendadas

---

## Base



1. Carretero 2020:
  1. Cap. 5
2. Carretero 2007:
  1. Cap. 3 y 7

## Recomendada



1. Tanenbaum 2006(en):
  1. Cap.3
2. Stallings 2005:
  1. 3.2, 3.3 y 3.5
3. Silberschatz 2006:
  1. 3.1 y 3.3

# ¡ATENCIÓN!

---

- ❑ Este material es un guión de la clase pero no son los apuntes de la asignatura.
- ❑ Los libros dados en la bibliografía junto con lo explicado en clase representa el material de estudio para el temario de la asignatura.



Don't forget that Linux became only possible because 20 years of OS research was carefully studied, analyzed, discussed and thrown away.

(Ingo Molnar)

izquotes.com

# Contenidos

---

## 1. Introducción

- Definición de proceso.
- Modelo ofrecido: recursos, multiprogramación, multitarea y multiproceso

## 2. Ciclo de vida del proceso: estado de procesos.

## 3. Servicios para gestionar procesos que da el sistema operativo.

## 4. Definición de hilo o *thread*

## 5. Hilos de biblioteca y núcleo.

## 6. Servicios para hilos en el sistema operativo.



# Contenidos

---

## 1. Introducción

- Definición de proceso.
- Modelo ofrecido: recursos, multiprogramación, multitarea y multiproceso

2. Ciclo de vida del proceso: estado de procesos.

3. Servicios para gestionar procesos que da el sistema operativo.

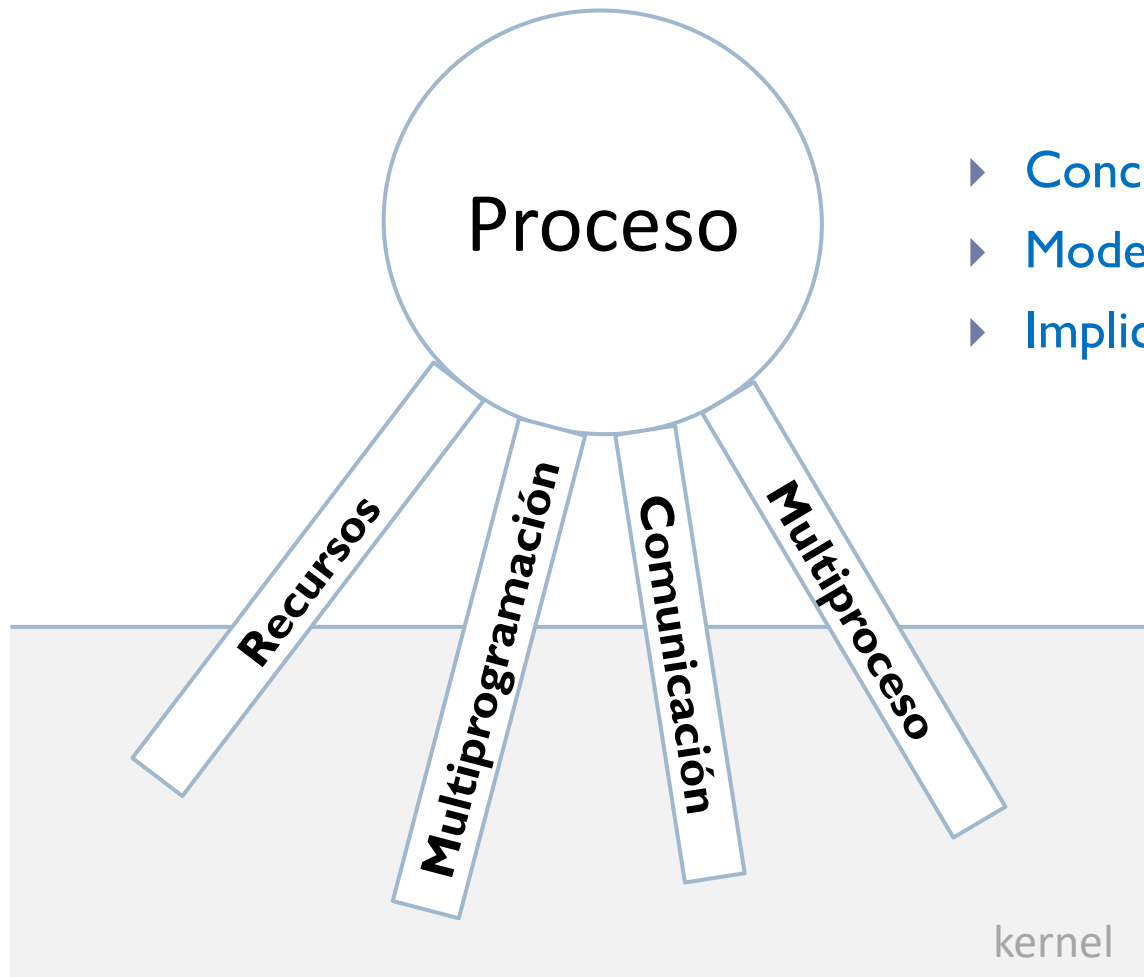
4. Definición de hilo o *thread*

5. Hilos de biblioteca y núcleo.

6. Servicios para hilos en el sistema operativo.

# Introducción

---



- ▶ Concepto de proceso
- ▶ Modelo ofrecido
- ▶ Implicaciones en S.O.

# Introducción

---



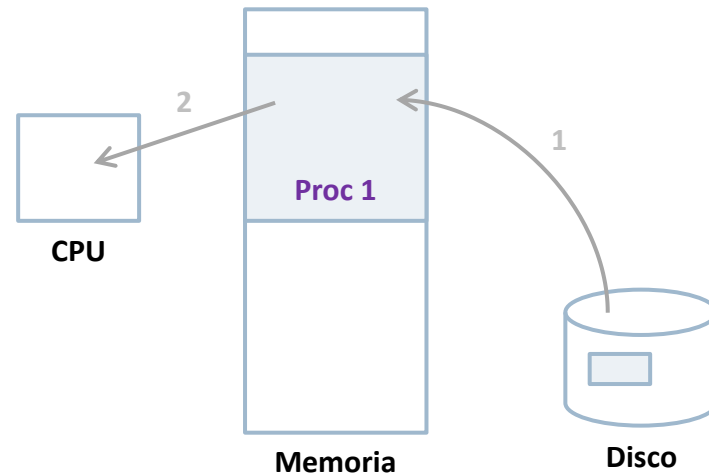
Proceso

► **Concepto de proceso**



# Concepto de proceso

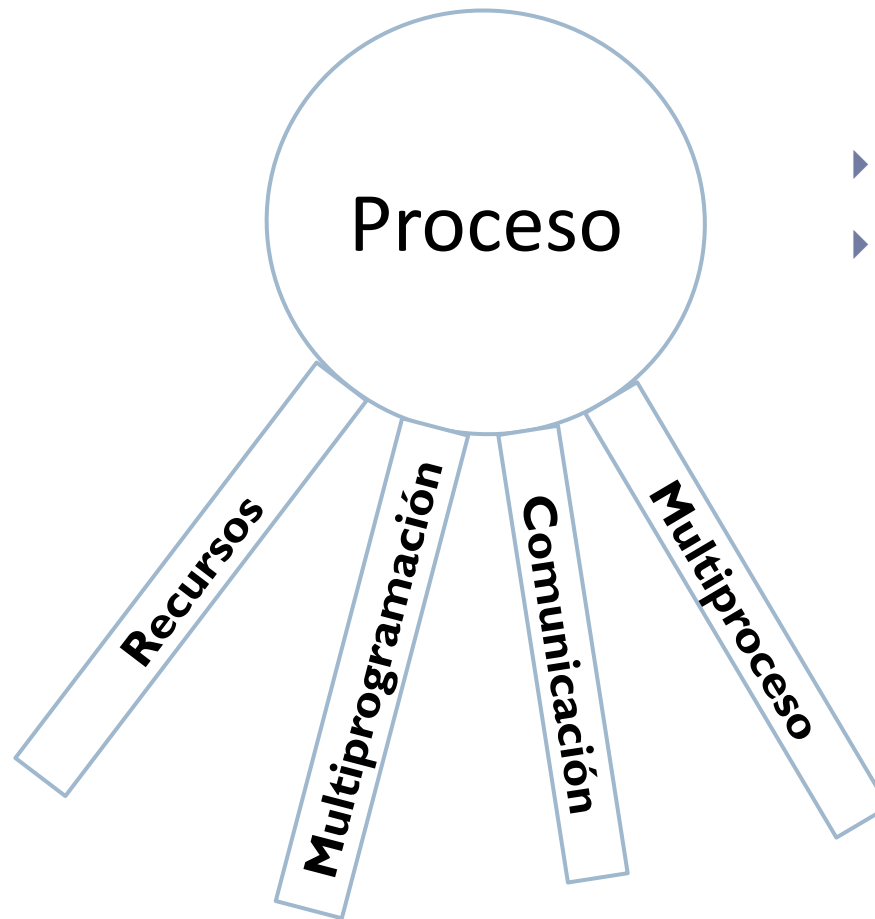
---



- ▶ **Proceso**
  - ▶ Programa en ejecución
  - ▶ Unidad de procesamiento gestionada por el S.O.

# Introducción

---



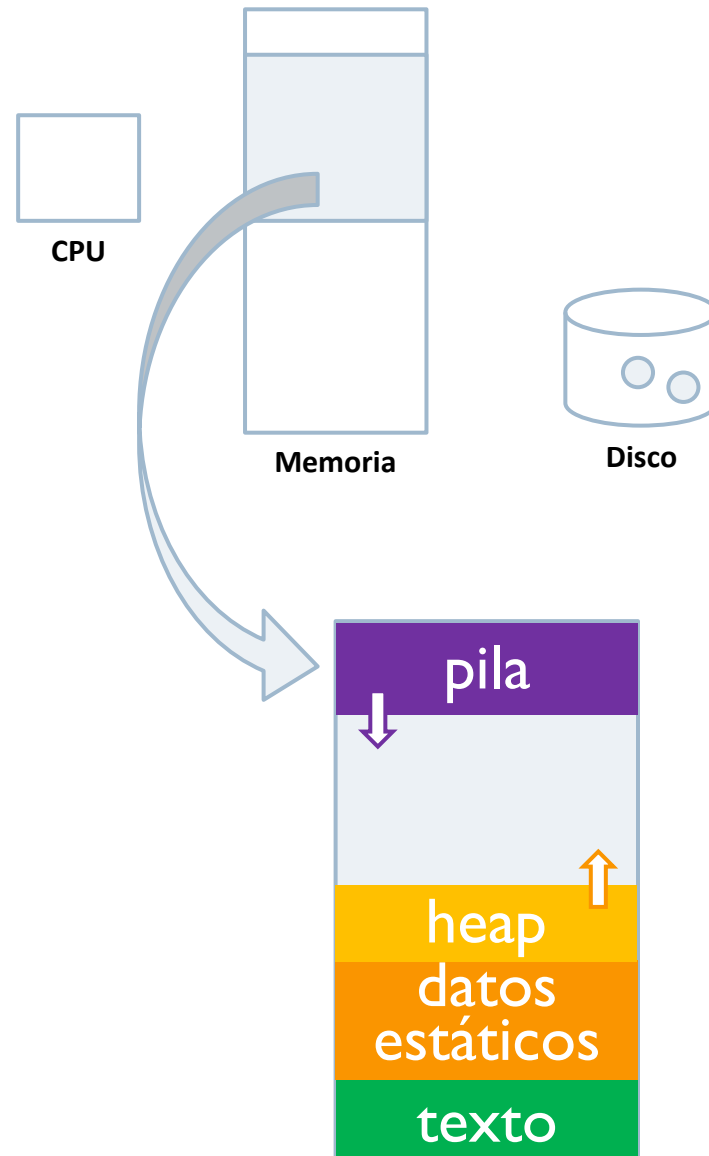
- ▶ Concepto de proceso
- ▶ **Modelo ofrecido**

# Modelo ofrecido

- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- multitarea
- multiproceso

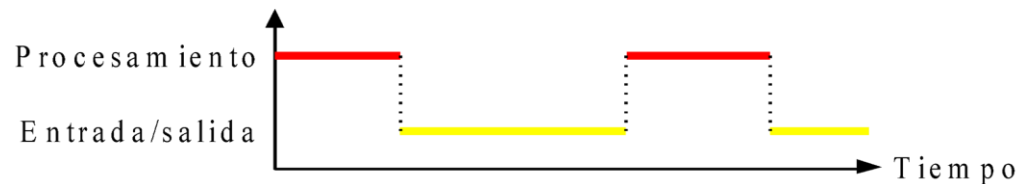
## ► Recursos asociados

- Zonas de memoria
  - Al menos: código, datos y pila
- Archivos abiertos
- Señales

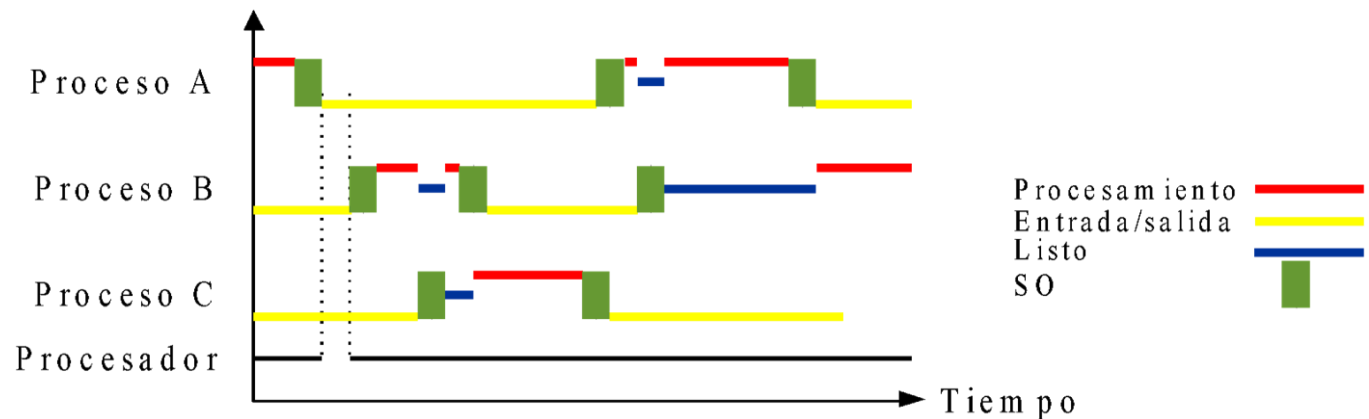


# Principios de la multitarea... (multiprogramación)

- ▶ Alternancia de fases de E/S y de procesamiento en los procesos:

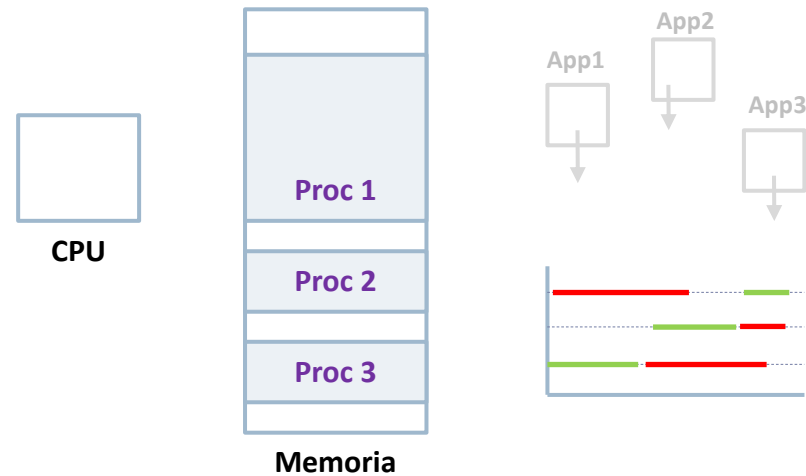


- ▶ La memoria almacena varios procesos.
- ▶ Paralelismo real entre E/S y CPU/UCP (DMA).



# Modelo ofrecido

- recursos
- **multiprogramación**
  - protección/compartición
  - jerarquía de procesos
- multitarea
- multiproceso

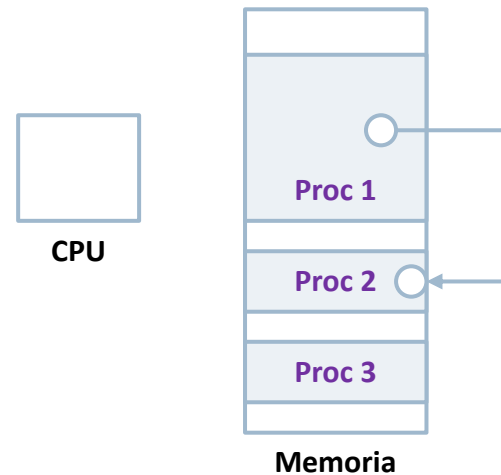


## ► Multiprogramación

- Tener varias aplicaciones en memoria
- Si una aplicación se bloquea por E/S, entonces se ejecuta mientras otra hasta que quede bloqueada
  - Cambio de contexto voluntario (C.C.V.)
- Eficiencia en el uso del procesador
- Grado de multiprogramación = número de aplicaciones en RAM

# Modelo ofrecido

- recursos
- multiprogramación
  - **protección/compartición**
  - jerarquía de procesos
- multitarea
- multiproceso

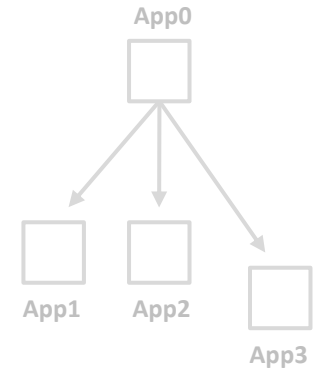
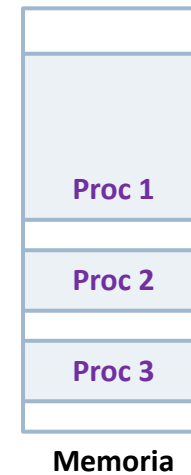


## ► Protección / Compartición

- El espacio de direcciones privado por aplicación, pero
- Posibilidad de comunicar datos entre dos aplicaciones
  - Paso de mensajes
  - Compartición de memoria

# Modelo ofrecido

- recursos
- multiprogramación
  - protección/compartición
  - **jerarquía de procesos**
- multitarea
- multiproceso



## ► Jerarquía de procesos

### ► Creación de proceso

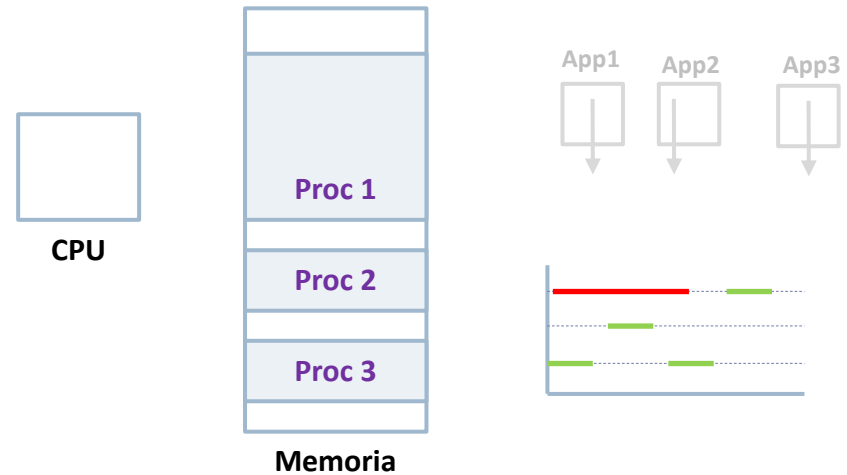
- Como copia de otro proceso existente
- A partir del programa en disco
- Como proceso en el arranque

### ► Grupo de procesos que pueden compartir mismo tratamiento

- Ej.: en VMS al terminar un proceso todos los hijos acaban (cascada)

# Modelo ofrecido

- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- **multitarea**
- multiproceso



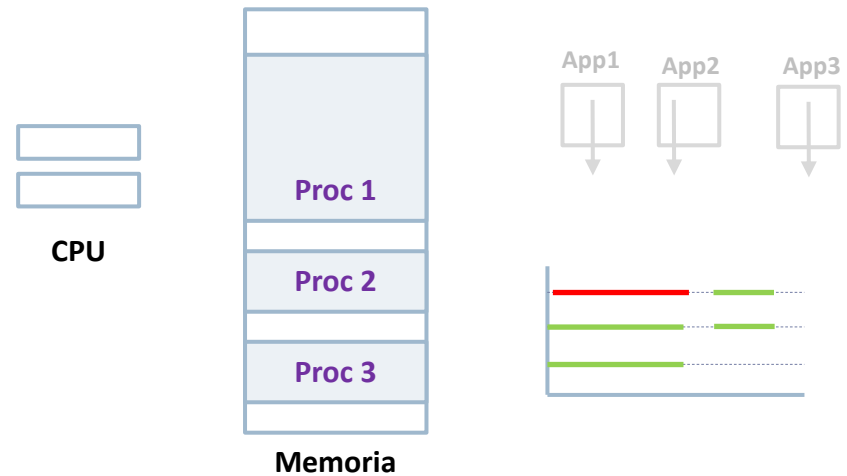
## ► Multitarea

- Cada proceso se ejecuta un quantum de tiempo (Ej.: 5 ms) y se rota el turno para ejecutar procesos no bloqueados
  - Cambio de contexto involuntario (C.C.I.)
- Reparto del uso del procesador
  - Parece que todo se ejecuta a la vez



# Modelo ofrecido

- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- multitarea
- **multiproceso**



## ► Multiproceso

- Se dispone de varios procesadores (multicore/multiprocesador)
- Además del reparto de cada CPU (multitarea) hay paralelismo real entre varias tareas (tantas como procesadores)
  - Se suele usar planificador y estructuras de datos separadas por procesador con algún mecanismo de equilibrio de carga

# Tipos de sistemas operativos

modelo a medida

---

## Sistemas Operativos

Multiproceso  
(varios procesos en ejecución)

Monoproceso  
(un único proceso)



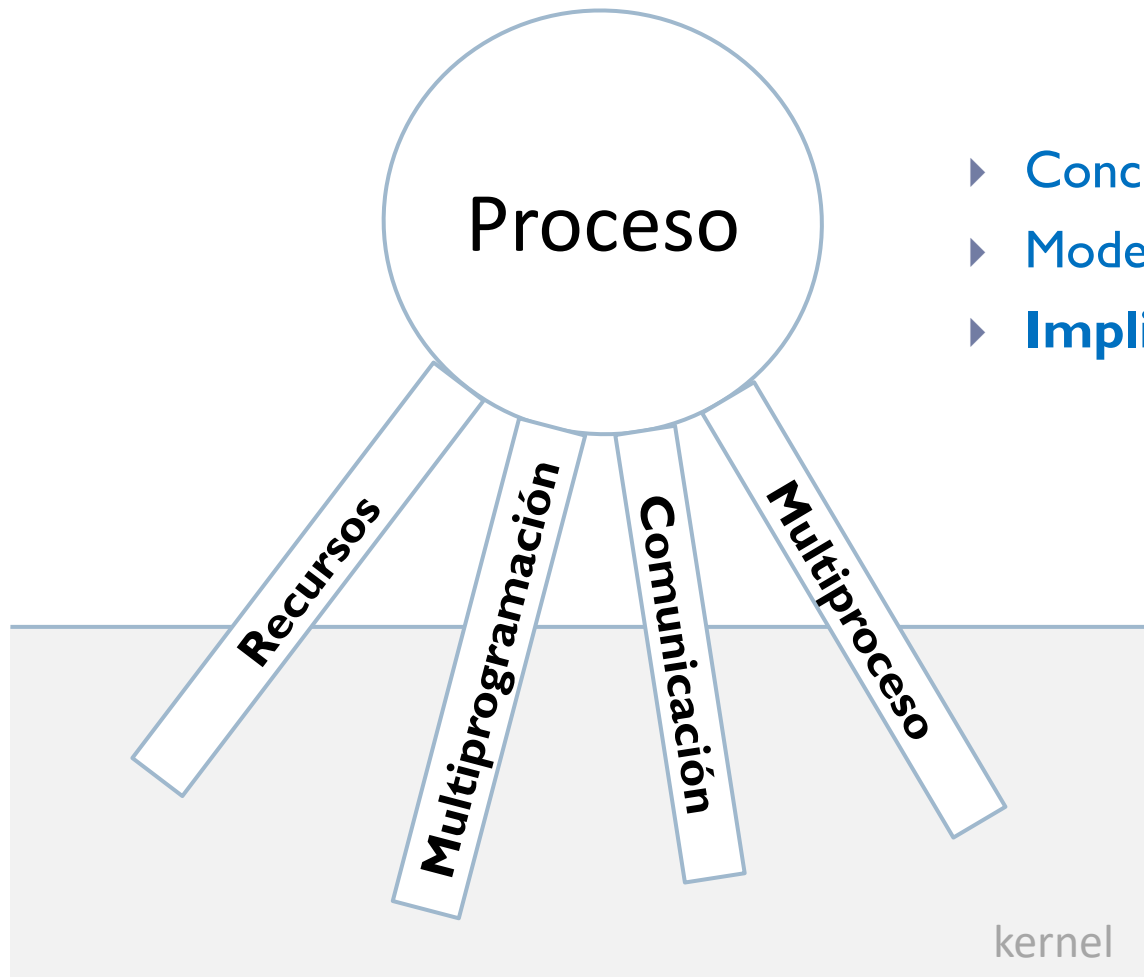
Multiusuario  
(varios usuarios  
a la vez)

Monousuario  
(un único  
usuario a la vez)

Monousuario  
(un único usuario a la vez)

# Introducción

---



- ▶ Concepto de proceso
- ▶ Modelo ofrecido
- ▶ **Implicaciones en S.O.**

# Implicaciones en el sistema operativo

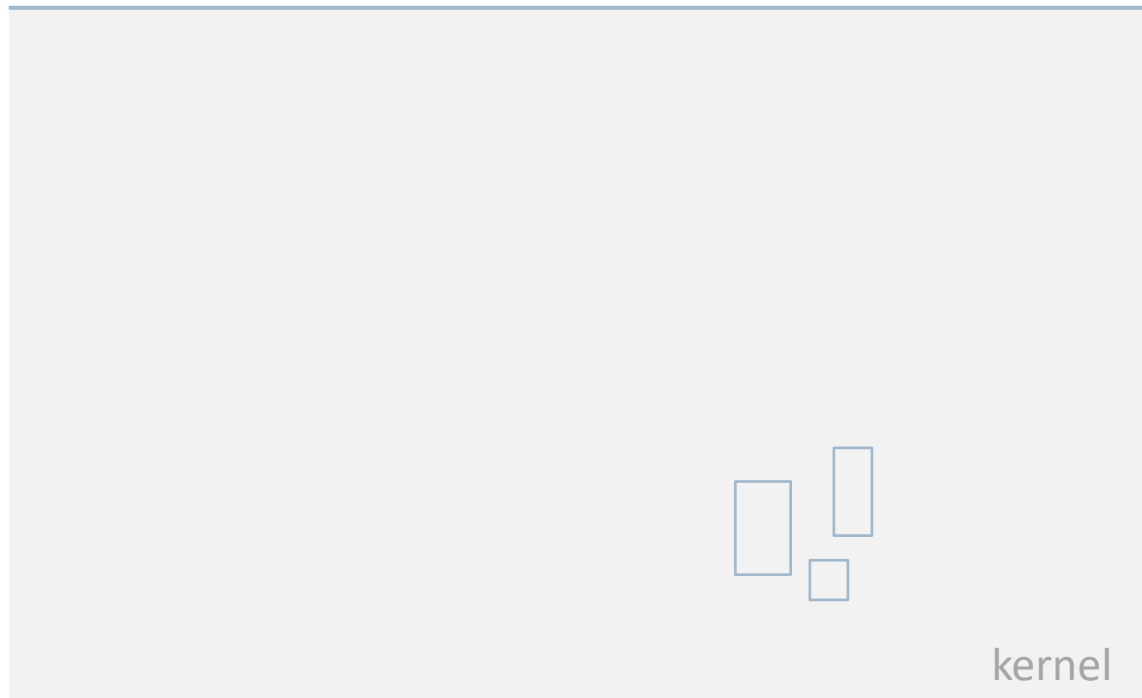
## I. Estructuras de datos

Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	<ul style="list-style-type: none"><li>• Zonas de memoria (código, datos y pila)</li><li>• Archivos abiertos</li><li>• Señales activas</li></ul>	<ul style="list-style-type: none"><li>• Diversas funciones internas</li><li>• Diversas funciones de servicio para memoria, ficheros, etc.</li></ul>
Multiprogramación	<ul style="list-style-type: none"><li>• Estado de ejecución</li><li>• Contexto: registros de CPU...</li><li>• Lista de procesos</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de dispositivos</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>
○ Protección / Compartición	<ul style="list-style-type: none"><li>• Paso de mensajes<ul style="list-style-type: none"><li>• Cola de mensajes de recepción</li></ul></li><li>• Memoria compartida<ul style="list-style-type: none"><li>• Zonas, locks y conditions</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Envío/Recepción mensaje y gestión de la cola de mensaje</li><li>• API concurrencia y gestión de estructuras de datos</li></ul>
○ Jerarquía de procesos	<ul style="list-style-type: none"><li>• Relación de parentesco</li><li>• Conjuntos de procesos relacionados</li><li>• Procesos de una misma sesión</li></ul>	<ul style="list-style-type: none"><li>• Clonar/Cambiar imagen de proceso</li><li>• Asociar procesos e indicar proceso representante</li></ul>
Multitarea	<ul style="list-style-type: none"><li>• Quantum restante</li><li>• Prioridad</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de reloj</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>
Multiproceso	<ul style="list-style-type: none"><li>• Afinidad</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de reloj</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>

# Implicaciones en el sistema operativo

---

## I. Estructuras de datos



# Implicaciones en el sistema operativo

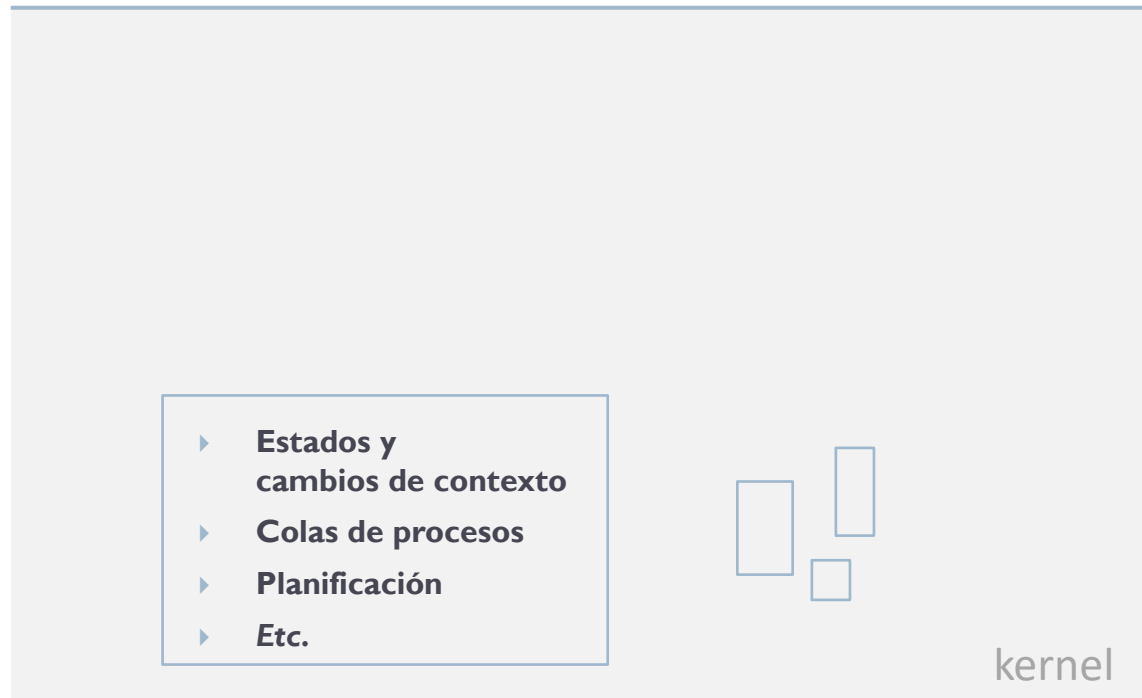
## 2. Funciones: de gestión internas

Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	<ul style="list-style-type: none"><li>• Zonas de memoria (código, datos y pila)</li><li>• Archivos abiertos</li><li>• Señales activas</li></ul>	<ul style="list-style-type: none"><li>• Diversas funciones internas</li><li>• Diversas funciones de servicio para memoria, ficheros, etc.</li></ul>
Multiprogramación	<ul style="list-style-type: none"><li>• Estado de ejecución</li><li>• Contexto: registros de CPU...</li><li>• Lista de procesos</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de dispositivos</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>
○ Protección / Compartición	<ul style="list-style-type: none"><li>• Paso de mensajes<ul style="list-style-type: none"><li>• Cola de mensajes de recepción</li></ul></li><li>• Memoria compartida<ul style="list-style-type: none"><li>• Zonas, locks y conditions</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Envío/Recepción mensaje y gestión de la cola de mensaje</li><li>• API concurrencia y gestión de estructuras de datos</li></ul>
○ Jerarquía de procesos	<ul style="list-style-type: none"><li>• Relación de parentesco</li><li>• Conjuntos de procesos relacionados</li><li>• Procesos de una misma sesión</li></ul>	<ul style="list-style-type: none"><li>• Clonar/Cambiar imagen de proceso</li><li>• Asociar procesos e indicar proceso representante</li></ul>
Multitarea	<ul style="list-style-type: none"><li>• Quantum restante</li><li>• Prioridad</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de reloj</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>
Multiproceso	<ul style="list-style-type: none"><li>• Afinidad</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de reloj</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>

# Implicaciones en el sistema operativo

---

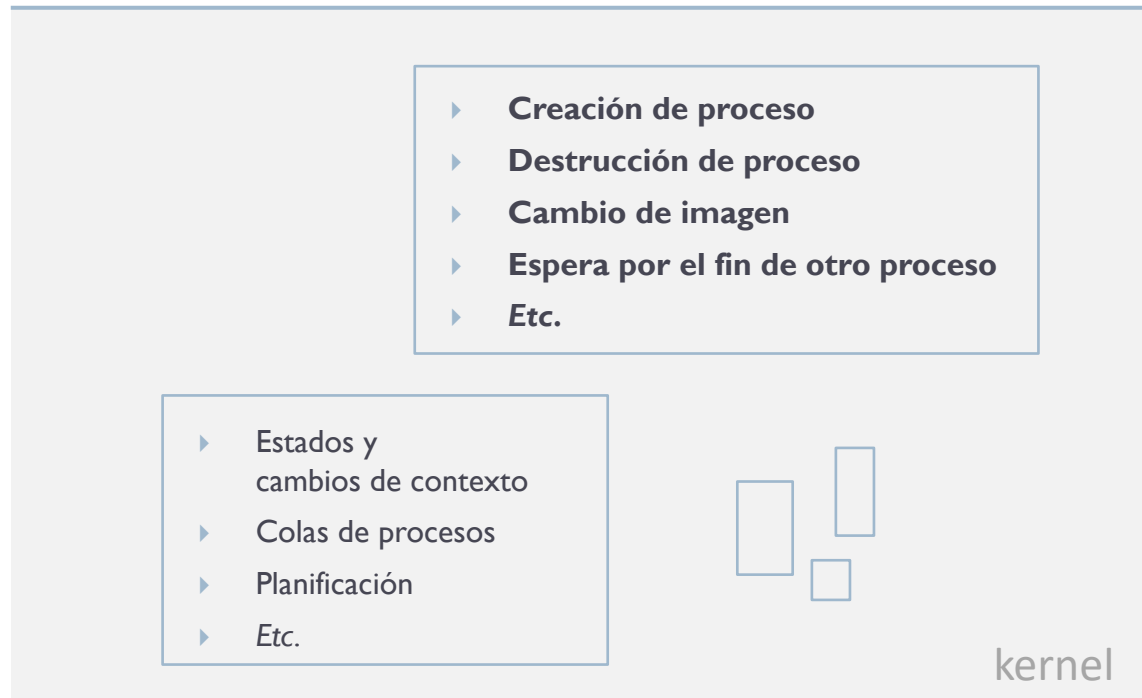
## 2. Funciones: de gestión internas



# Implicaciones en el sistema operativo

---

## 3. Funciones: de servicio





# Implicaciones en el sistema operativo

---

## 3. Funciones:API de servicio

- ▶ fork, exit, exec, wait, ...
- ▶ *pthread\_create, pthread...*

- ▶ Creación de proceso
- ▶ Destrucción de proceso
- ▶ Cambio de imagen
- ▶ Espera por el fin de otro proceso
- ▶ *Etc.*

- ▶ Estados y cambios de contexto
- ▶ Colas de procesos
- ▶ Planificación
- ▶ *Etc.*

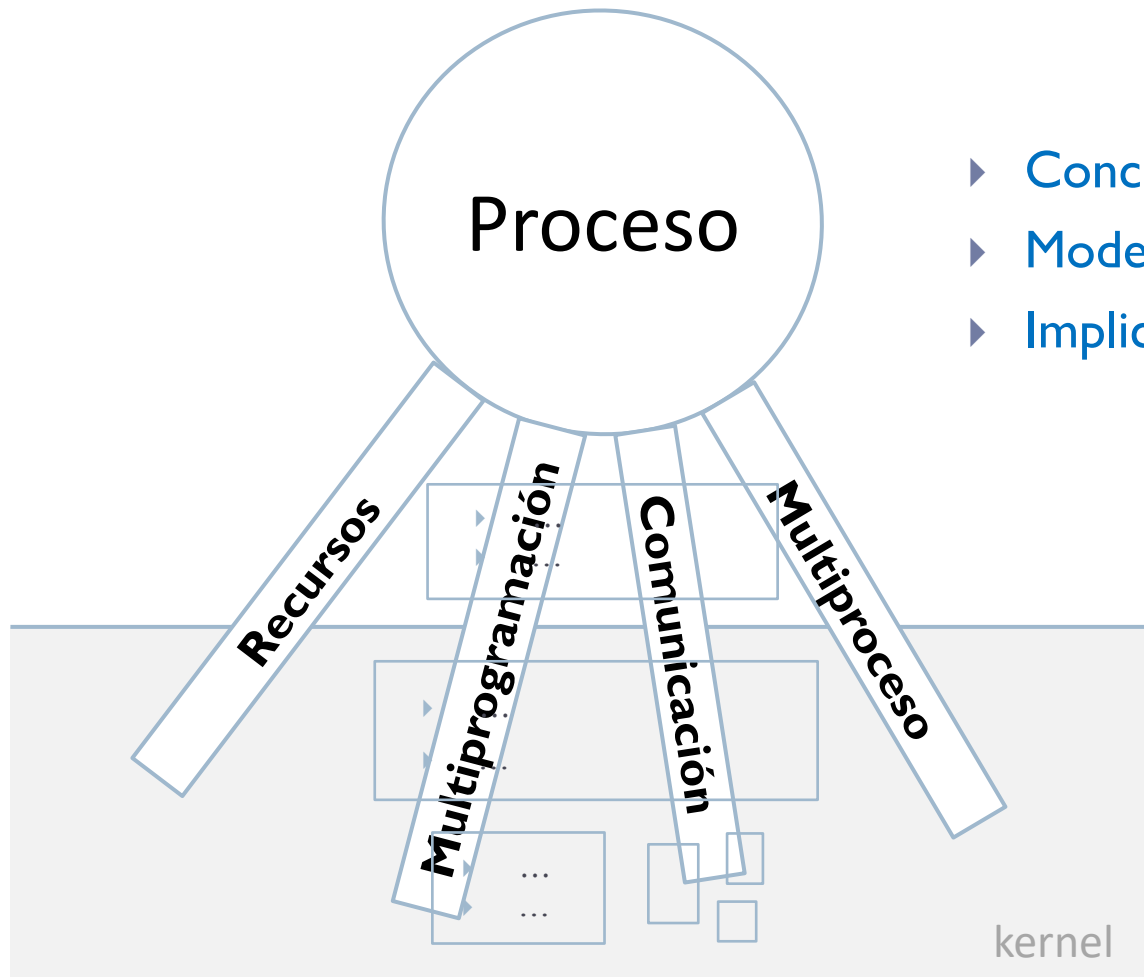


kernel

# Introducción

## resumen

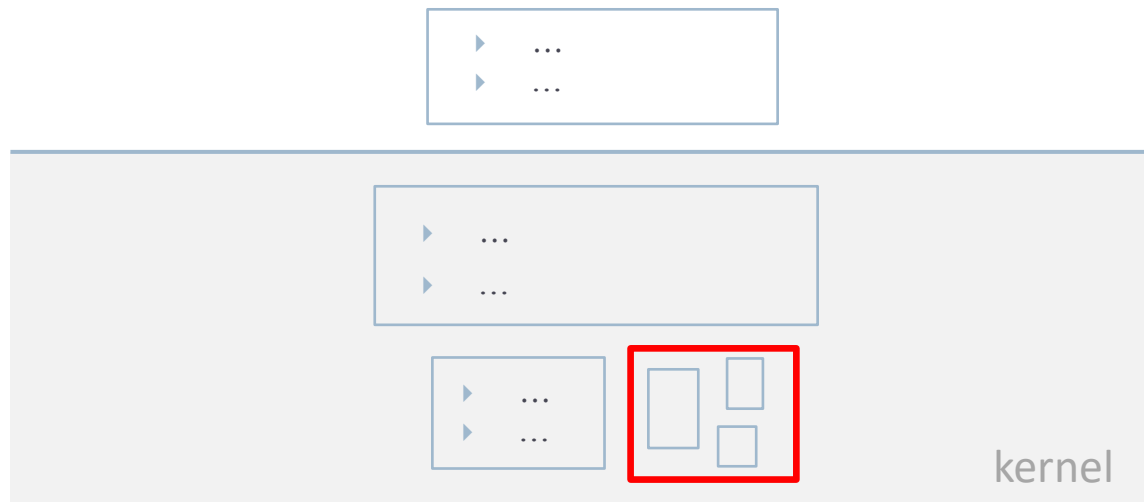
---



- ▶ Concepto de proceso
- ▶ Modelo ofrecido
- ▶ Implicaciones en S.O.

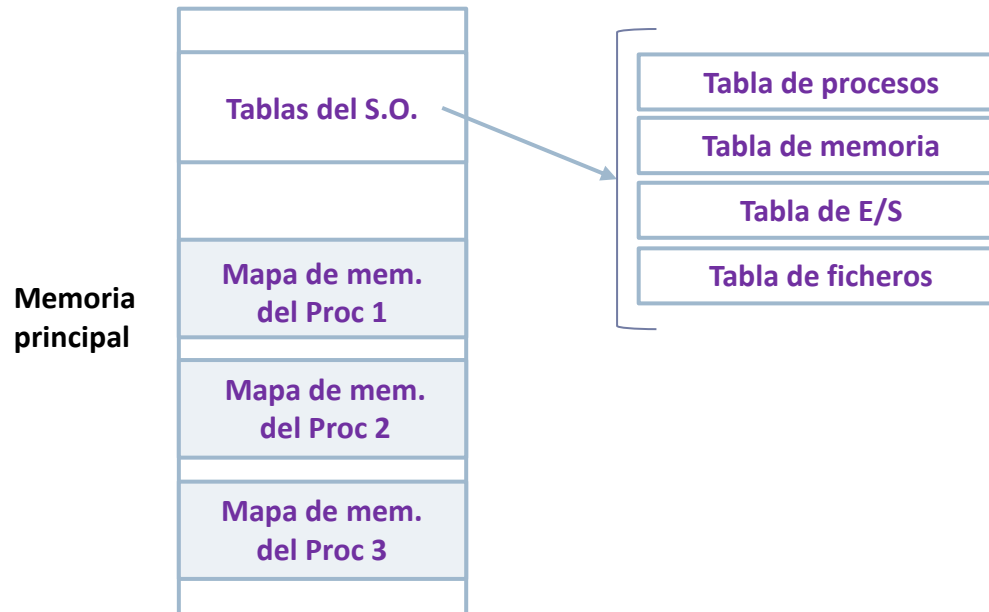
# Principales estructuras de datos

---



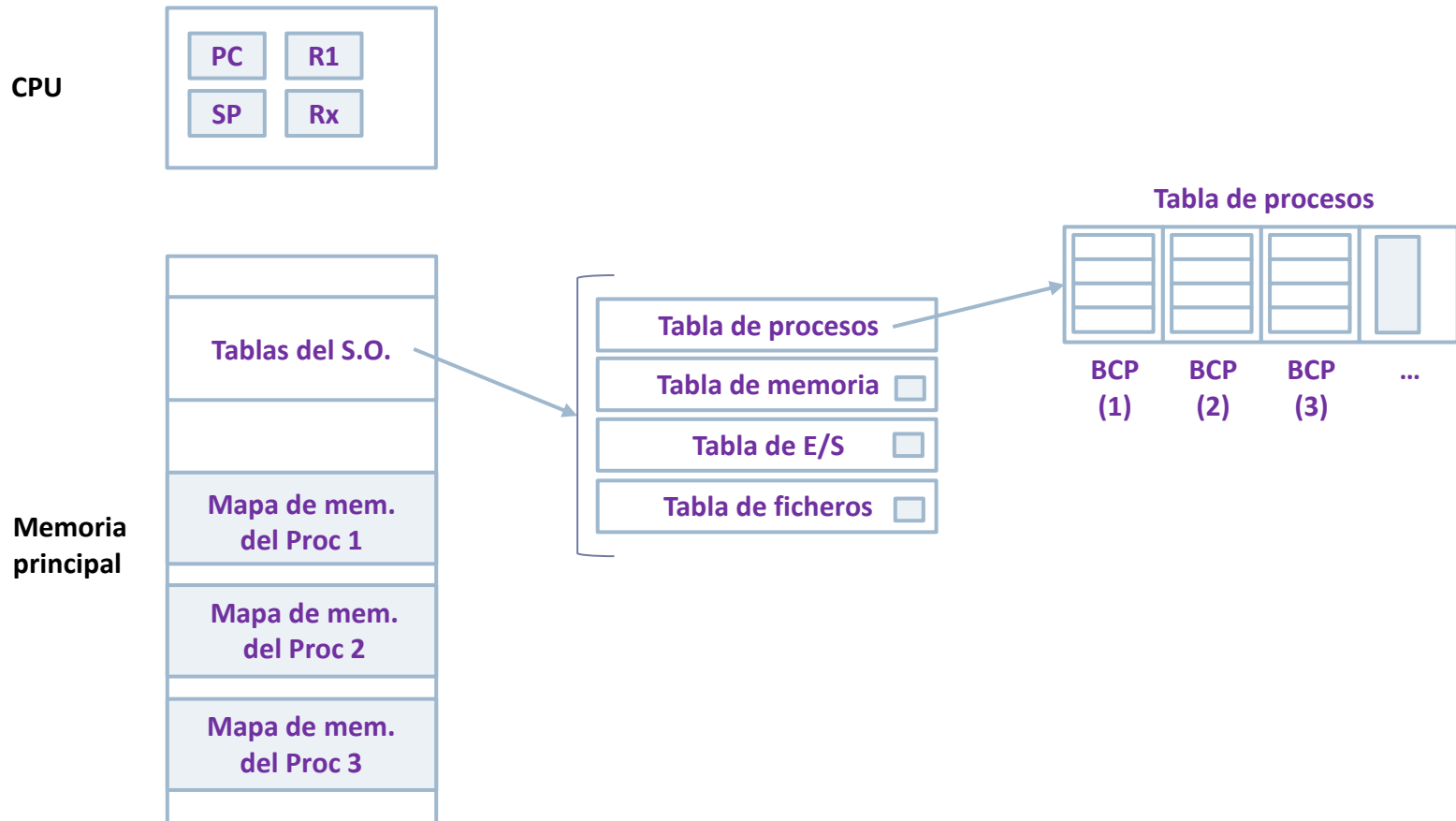
# Información en el sistema operativo

---



# Información para un proceso está: **en procesador + en memoria + datos adicionales del S.O.**

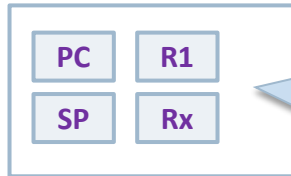
---



# Información para un proceso

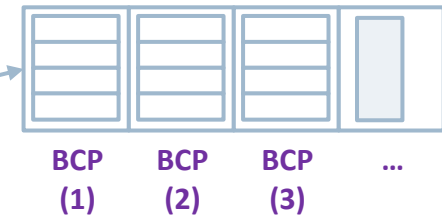
## estado del procesador

CPU

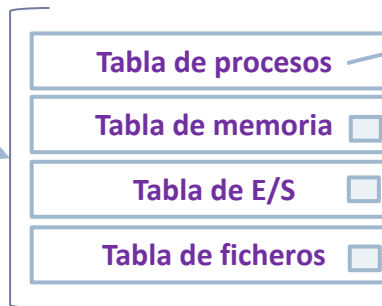
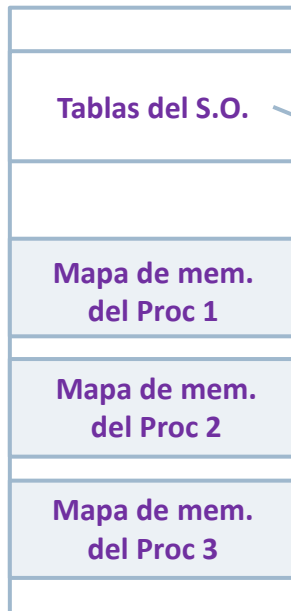


- **Estado del procesador** incluye todos los valores de los registros del procesador (accesibles por programador: PC, SP, ... y accesibles por el sistema operativo: RE, control de memoria, ...)
- Cambio de contexto = guardar estado proceso saliente + restaurar estado del proceso entrante a la CPU

Tabla de procesos

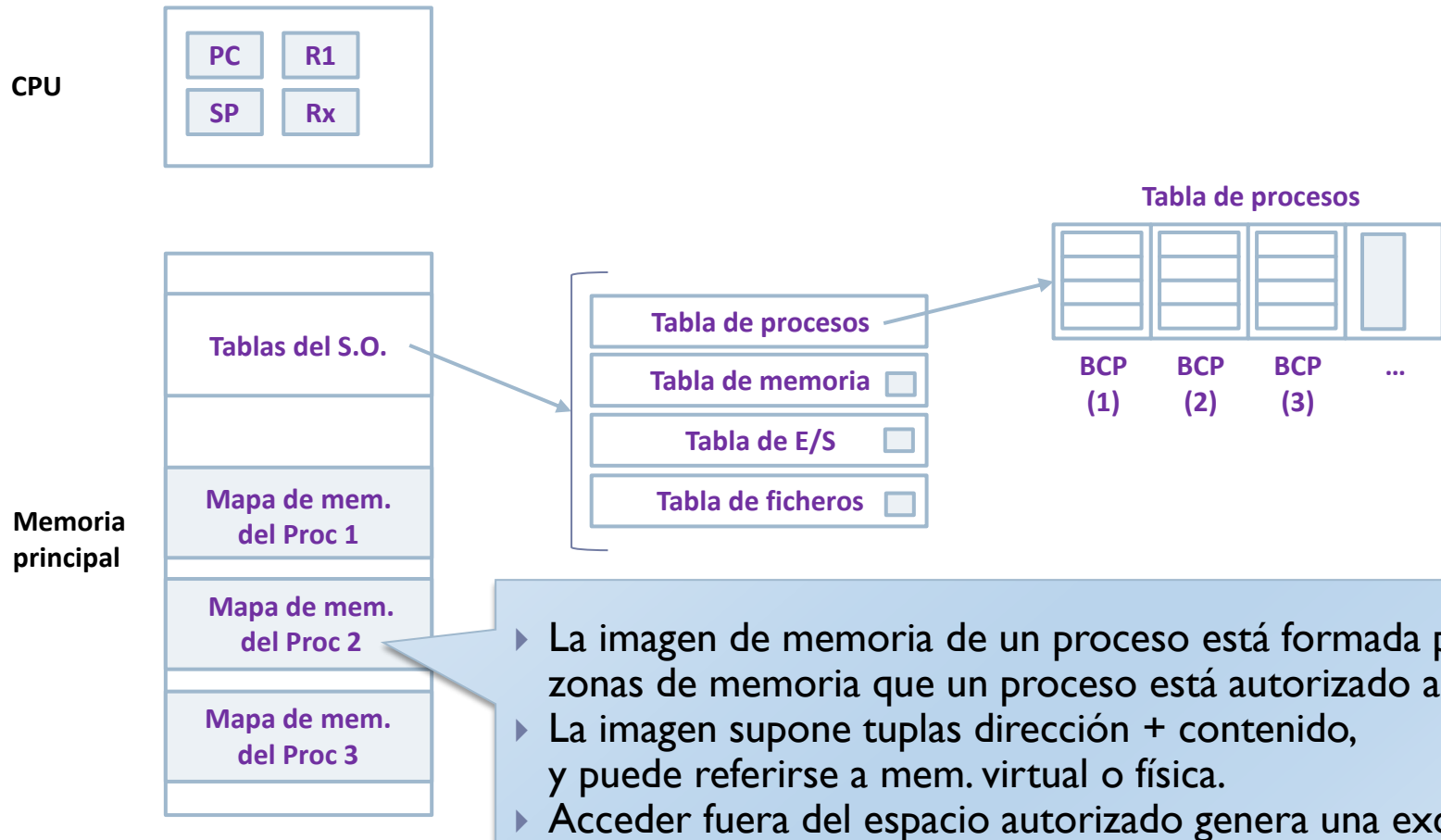


Memoria principal



# Información para un proceso

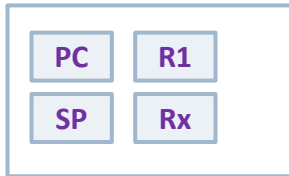
## imagen de memoria



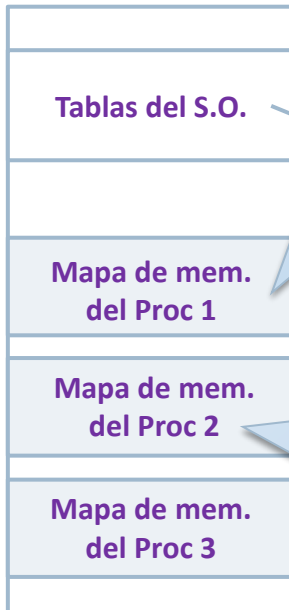
# Información para un proceso

## imagen de memoria

CPU



Memoria principal



	... de tamaño fijo	... tamaño variable
Proceso con única región...	Usado en sistemas sin memoria virtual	<ul style="list-style-type: none"> <li>▶ Sistemas sin memoria virtual:                             <ul style="list-style-type: none"> <li>▶ Necesita espacio de reserva → Desperdicio de memoria.</li> </ul> </li> <li>▶ Sistemas con memoria virtual:                             <ul style="list-style-type: none"> <li>▶ Espacio de reserva virtual → Factible, menos flexible que múltiples regiones.</li> </ul> </li> </ul>
Proceso con número fijo de regiones...		<ul style="list-style-type: none"> <li>▶ Regiones prefijadas (texto, datos, pila).</li> <li>▶ Cada región puede crecer.</li> <li>▶ Con memoria virtual el hueco entre pila y datos no consume recursos físicos</li> </ul>
Proceso con número variable de regiones...		<ul style="list-style-type: none"> <li>▶ Un proceso se estructura en un número arbitrario de regiones (más actual).</li> <li>▶ Más avanzado y muy flexible:                             <ul style="list-style-type: none"> <li>▶ Regiones compartidas.</li> <li>▶ Regiones con distintos permisos.</li> </ul> </li> </ul>

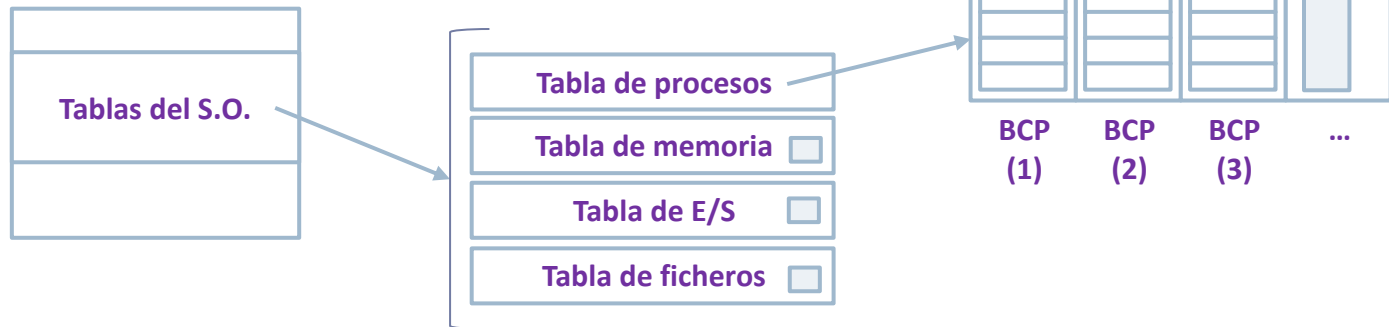
- ▶ La imagen de memoria de un proceso está formada por las zonas de memoria que un proceso está autorizado a usar.
- ▶ La imagen supone tuplas dirección + contenido, y puede referirse a mem. virtual o física.
- ▶ Acceder fuera del espacio autorizado genera una excepción.



# Información para un proceso

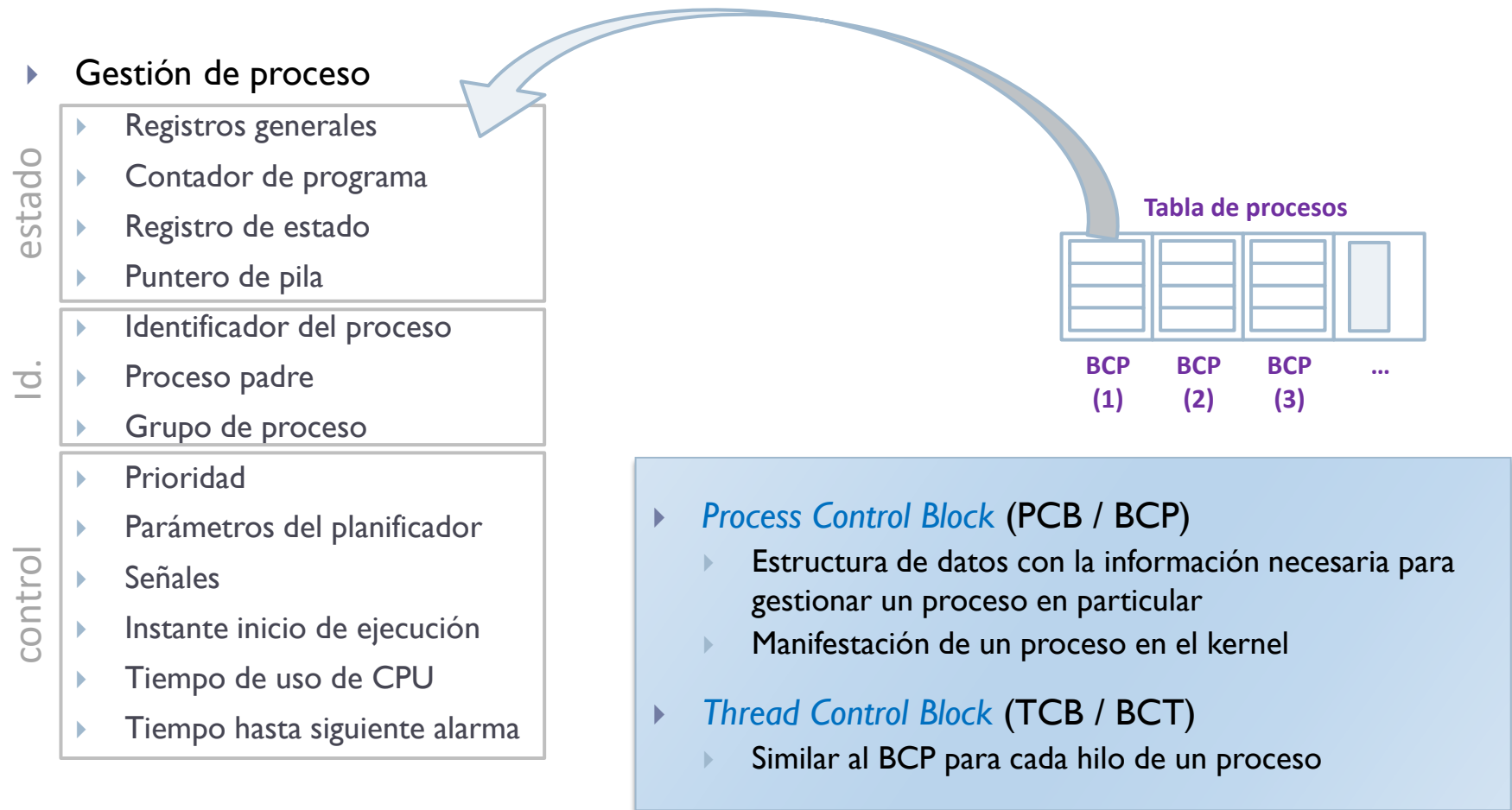
## datos gestionados por el S.O.

- ▶ La información de cada proceso está en el BCP...
- ▶ ...Información fuera del BCP:
  - ▶ Por razones de eficiencia
  - ▶ Para compartir información entre procesos

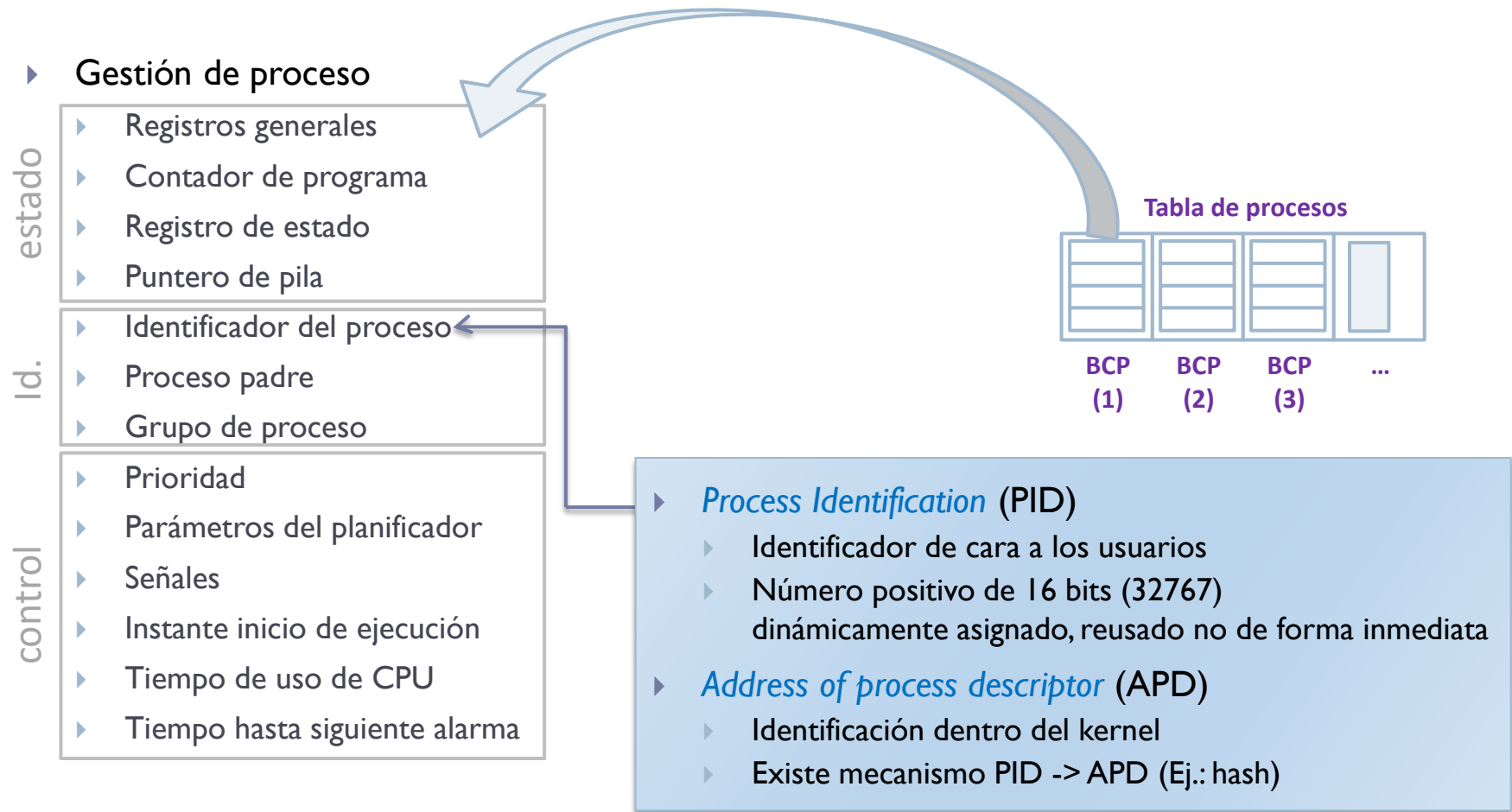


- ▶ Ejemplos:
  - ▶ Tabla de segmentos y páginas de **memoria**
  - ▶ Tabla de punteros de posición de **ficheros**
  - ▶ Lista de peticiones a **dispositivos**

# BCP: entrada de la tabla de procesos

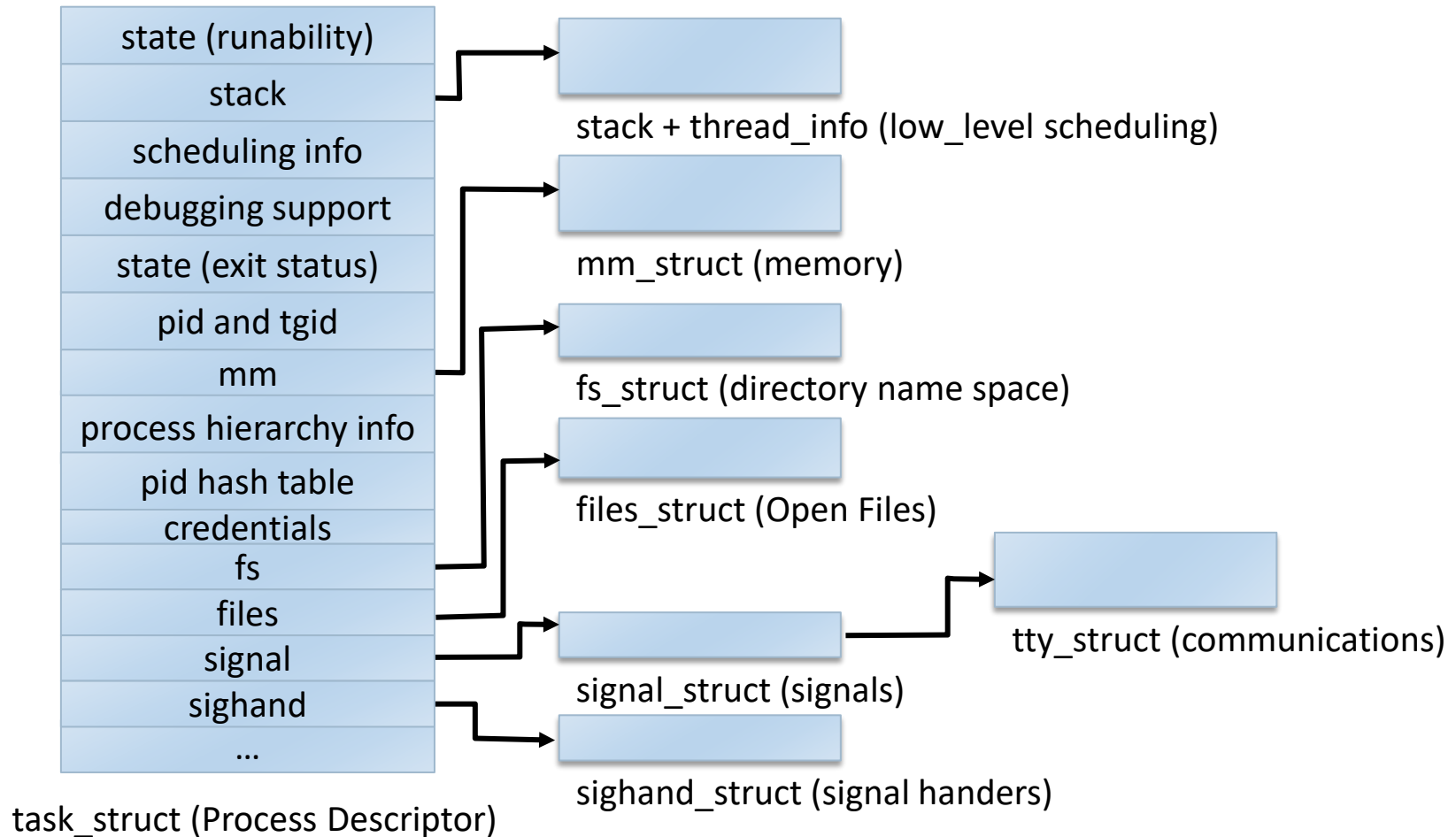


# BCP: entrada de la tabla de procesos



# Información del proceso

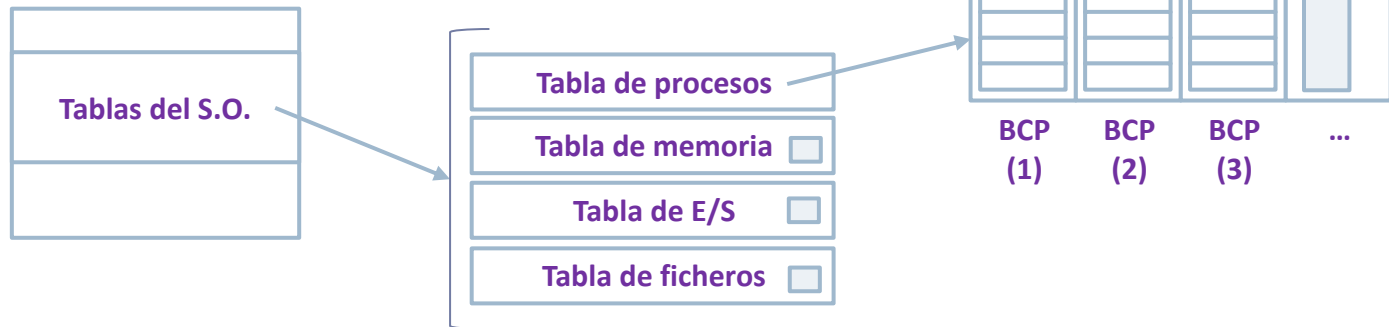
## Linux



# Información para un proceso

## datos gestionados por el S.O.

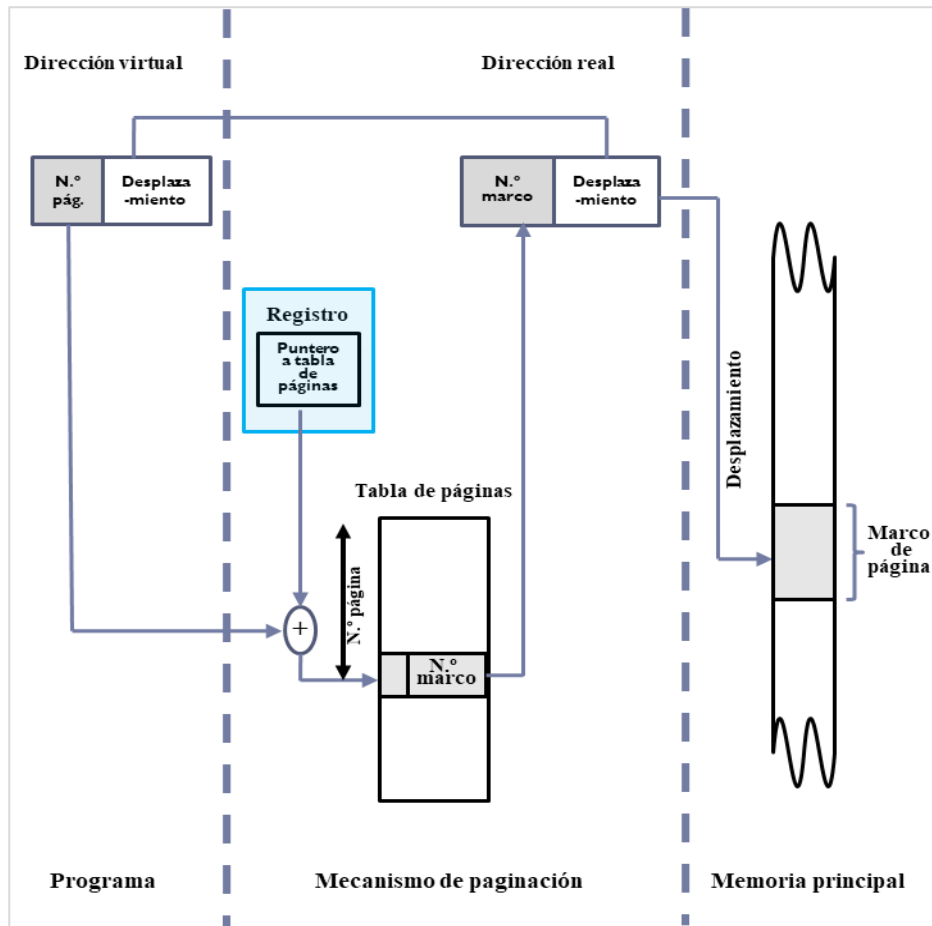
- ▶ La información de cada proceso está en el BCP...
- ▶ ...Información fuera del BCP:
  - ▶ Por razones de eficiencia
  - ▶ Para compartir información entre procesos



- ▶ Ejemplos:
  - ▶ **Tabla de segmentos y páginas de memoria**
  - ▶ **Tabla de punteros de posición de ficheros**
  - ▶ Lista de peticiones a dispositivos

# Información para un proceso

## datos gestionados por el S.O. => fuera del BCP

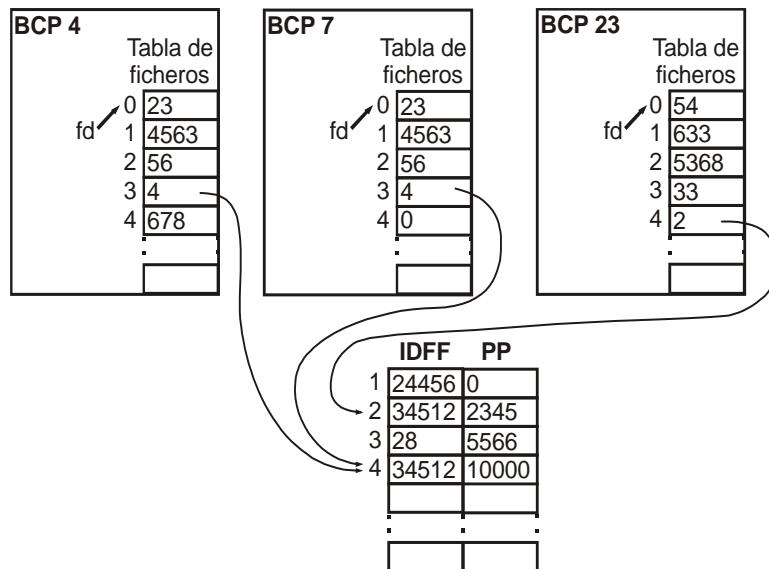


- ▶ **Tabla de páginas:**
  - ▶ Describe la imagen de memoria del proceso.
  - ▶ Razones:
    - ▶ Tiene tamaño variable.
    - ▶ La compartición de memoria entre procesos requiere se sea externa al BCP.
  - ▶ El BCP contiene el puntero a la tabla de páginas.

# Información para un proceso

## datos gestionados por el S.O. => fuera del BCP

- ▶ Tabla de punteros de posición de ficheros:



- ▶ Describe la posición de lectura/escritura de los ficheros abiertos.
- ▶ La compartición de estado del fichero entre procesos obliga a que sea externa al BCP.
- ▶ El BCP contiene el índice del elemento de la tabla que contiene la información del fichero abierto: el i-nodo y la posición de lectura/escritura.

# Contenidos

---

## 1. Introducción

- Definición de proceso.
- Modelo ofrecido: recursos, multiprogramación, multitarea y multiproceso

## 2. **Ciclo de vida del proceso: estado de procesos.**

## 3. Servicios para gestionar procesos que da el sistema operativo.

## 4. Definición de hilo o *thread*

## 5. Hilos de biblioteca y núcleo.

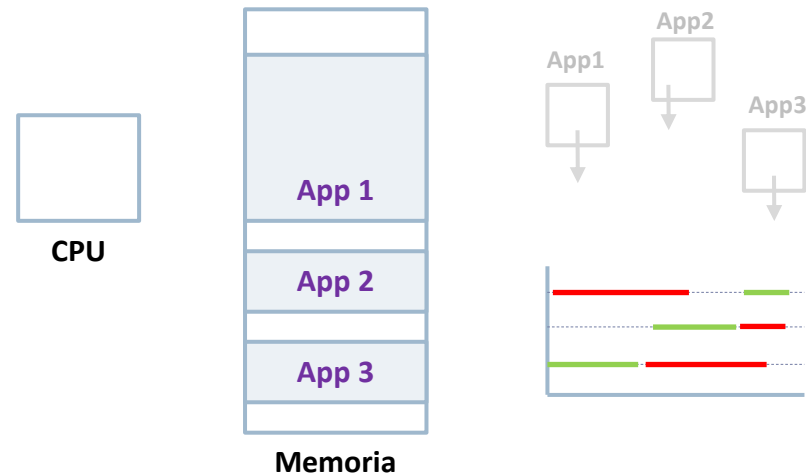
## 6. Servicios para hilos en el sistema operativo.



# Modelo ofrecido

## repaso

- recursos
- **multiprogramación**
  - protección/compartición
  - jerarquía de procesos
- multitarea
- multiproceso

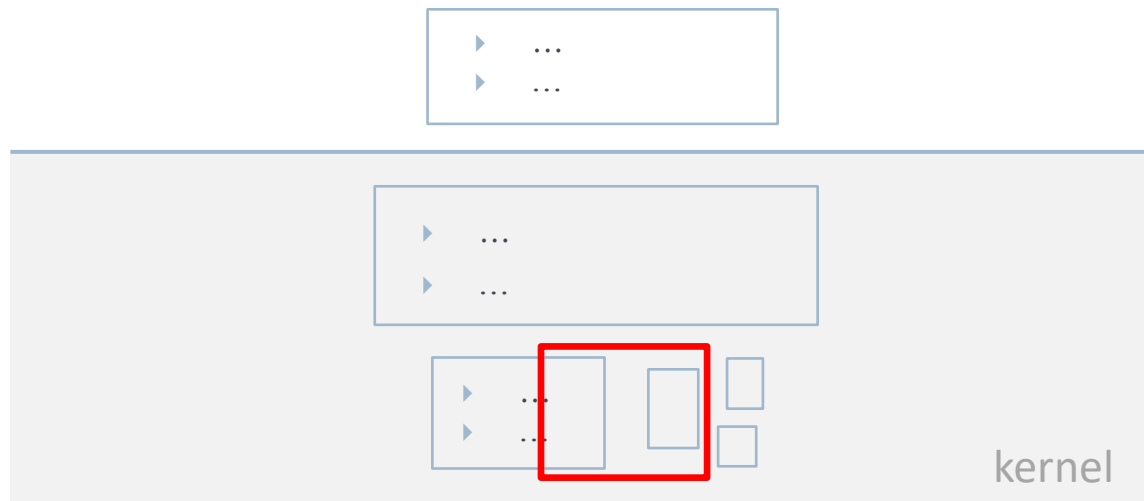


## ► Multiprogramación

- Tener varias aplicaciones en memoria
- Si una aplicación se bloquea por E/S, entonces se ejecuta mientras otra hasta que quede bloqueada
  - Cambio de contexto voluntario (C.C.V.)
- Eficiencia en el uso del procesador
- Grado de multiprogramación = número de aplicaciones en RAM

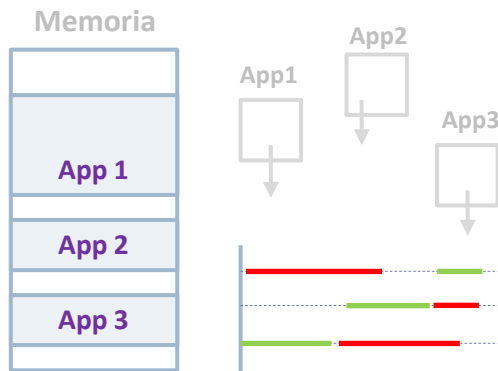
# Multiprogramación (datos y funciones)

Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Multiprogramación	<ul style="list-style-type: none"><li>• Estado de ejecución</li><li>• Contexto: registros de CPU...</li><li>• Lista de procesos</li></ul>	<ul style="list-style-type: none"><li>• Int. hw/sw de dispositivos</li><li>• Planificador</li><li>• Crear/Destruir/Planificar proceso</li></ul>



# Multiprogramación

---

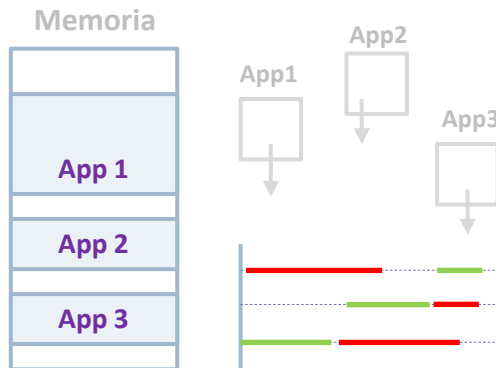
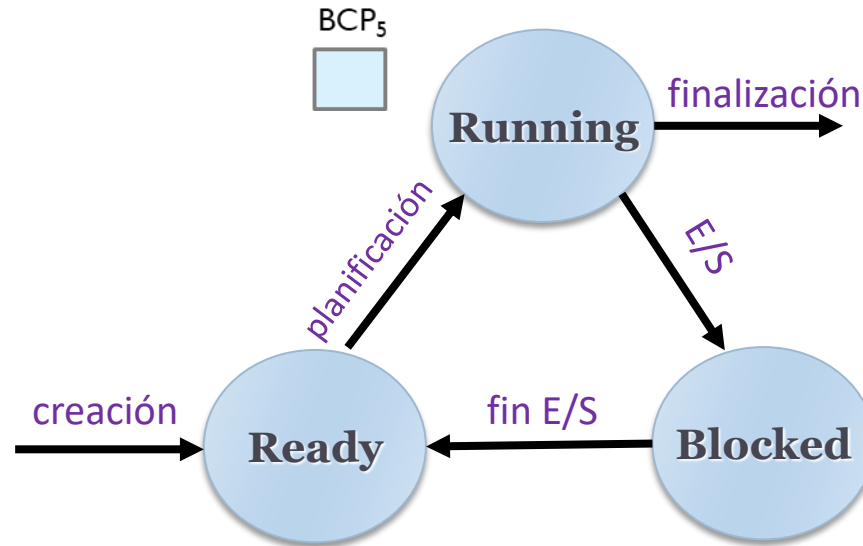


- ▶ Tener varias aplicaciones en memoria
- ▶ Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - ▶ Cambio de contexto voluntario (C.C.V.)

# Multiprogramación (datos)

## Estados de un proceso (c.c.v.)

- Estado
- Lista/Cola
- Contexto

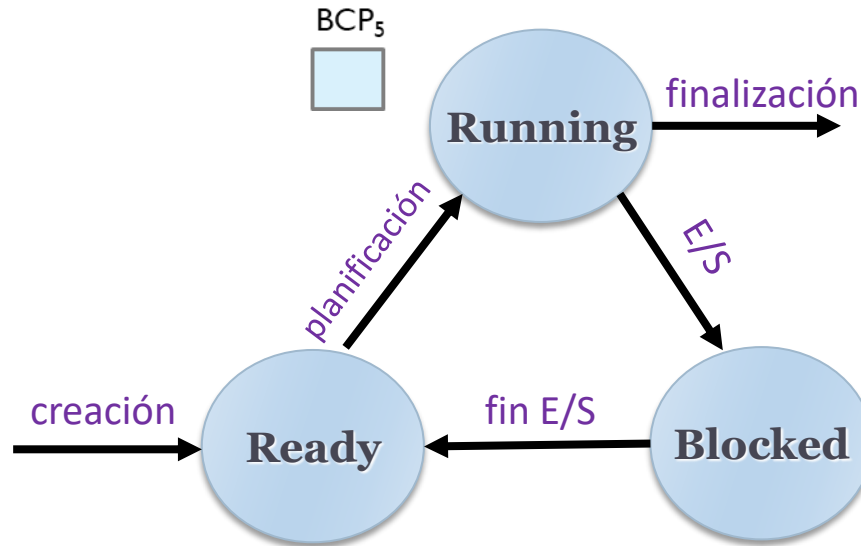


- ▶ Tener varias aplicaciones en memoria
- ▶ Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - ▶ Cambio de contexto voluntario (C.C.V.)

# Multiprogramación (datos)

## Estados de un proceso (c.c.v.)

- Estado
- Lista/Cola
- Contexto

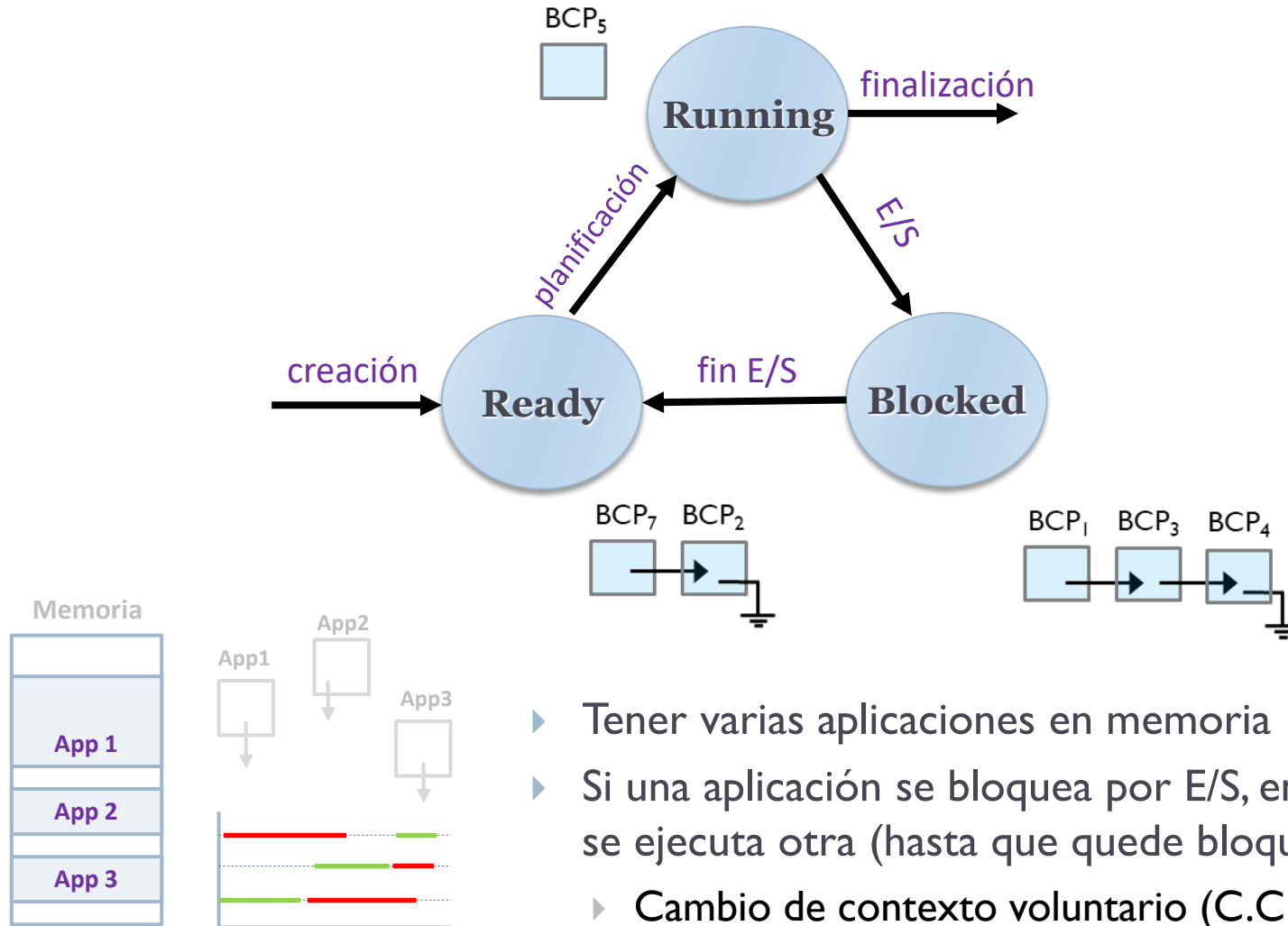


- ▶ **En ejecución:** con CPU asignada
- ▶ **Listo para ejecutar:** no procesador disponible para él
- ▶ **Bloqueado:** esperando un evento
- ▶ **Suspendido y listo:** expulsado pero listo para ejecutar
- ▶ **Suspendido y bloqueado:** expulsado y esperando evento

# Multiprogramación (datos)

## Lista/Colas de procesos (c.c.v.)

- Estado
- Lista/Cola
- Contexto

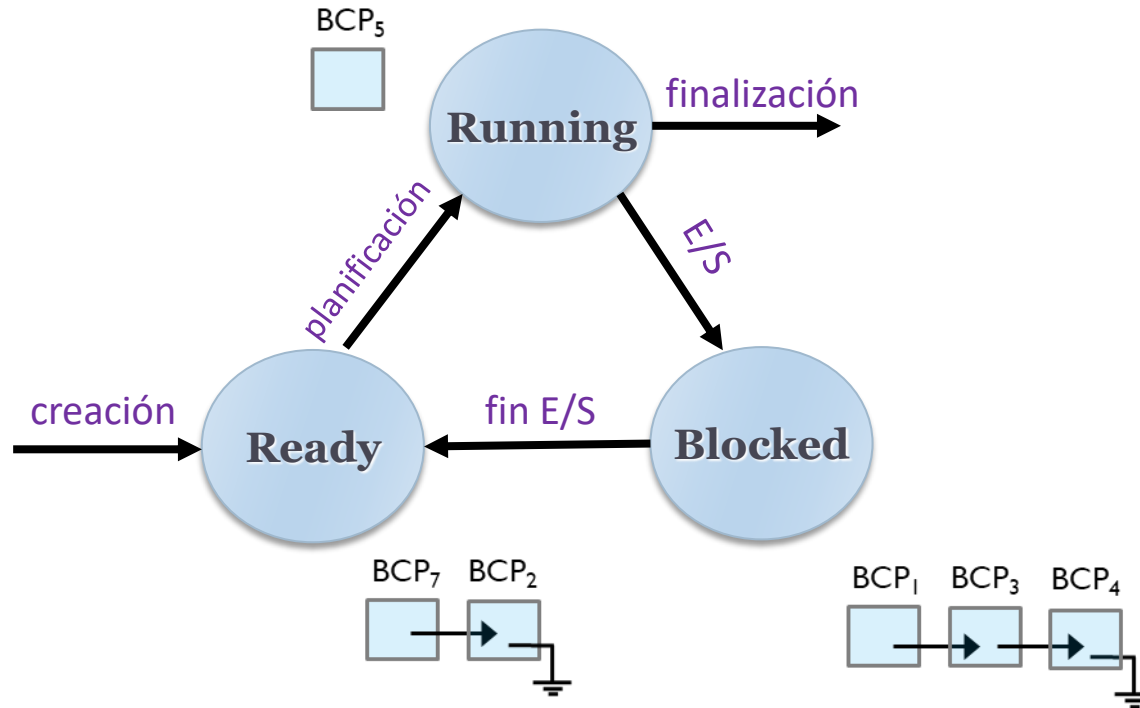


- ▶ Tener varias aplicaciones en memoria
- ▶ Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - ▶ Cambio de contexto voluntario (C.C.V.)

# Multiprogramación (datos)

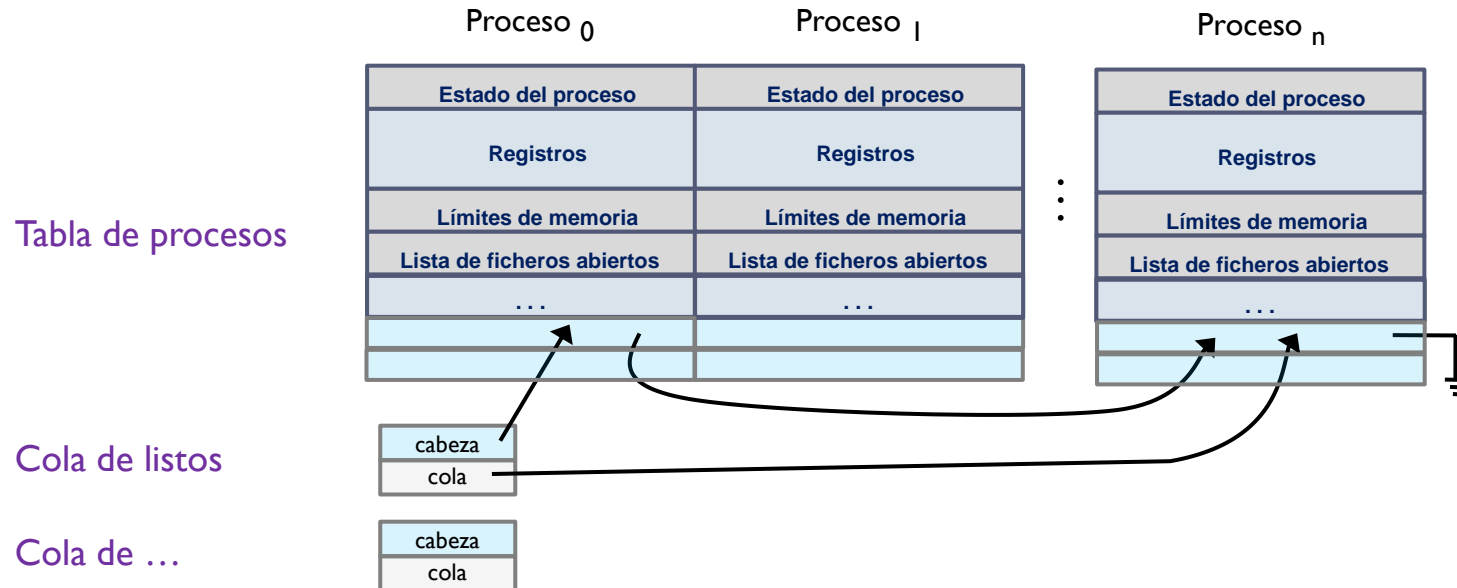
## Lista/Colas de procesos (c.c.v.)

- Estado
- Lista/Cola
- Contexto



- ▶ **Cola de listos:** procesos esperando a ejecutar en CPU
- ▶ **Cola de bloqueados por recurso:** procesos a la espera de finalizar una petición bloqueante al recurso asociado
- **Un proceso solo puede estar en una cola (como mucho)**

# Implementación de las colas de procesos



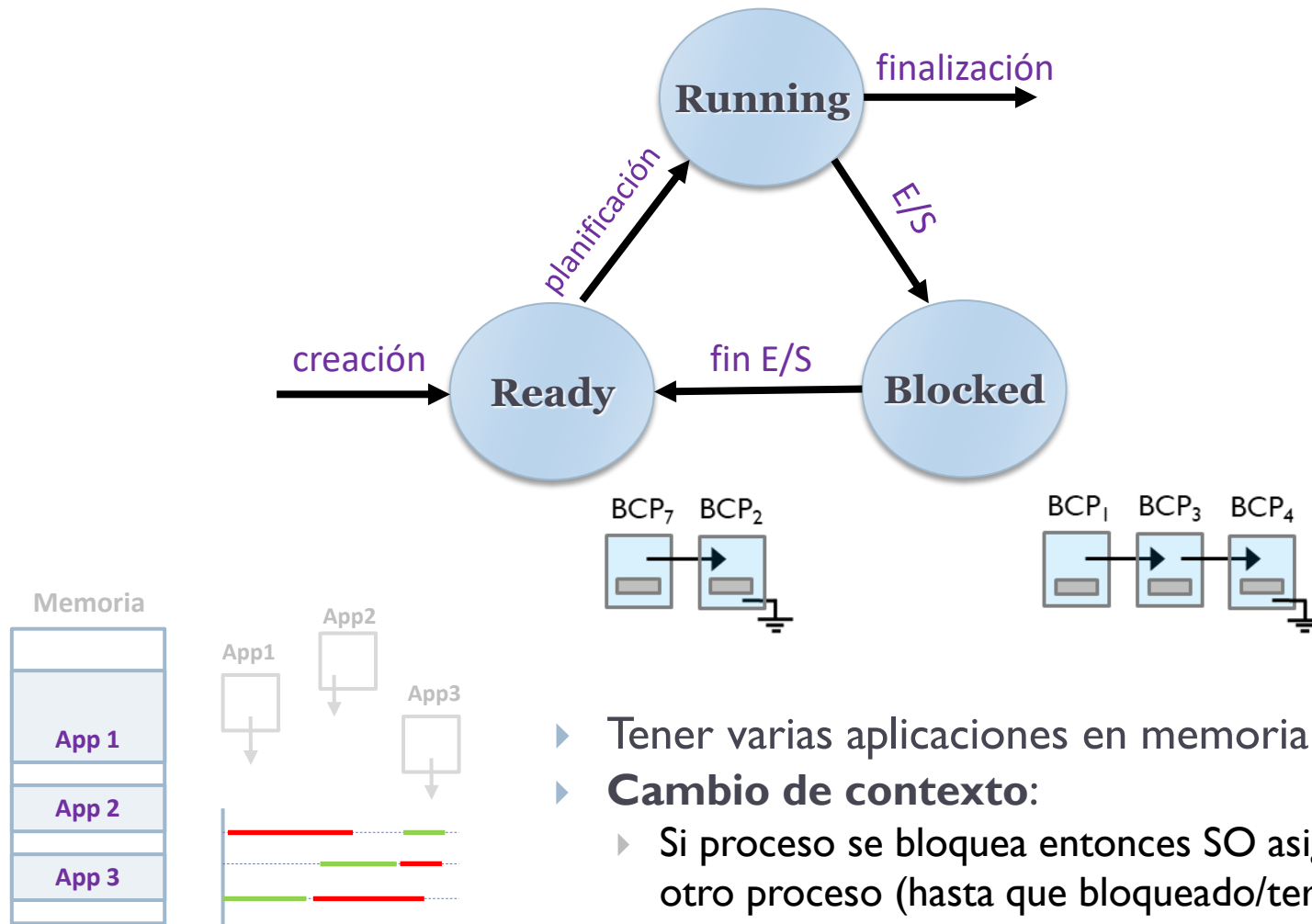
- ▶ Cola de listos: procesos esperando a ejecutar en CPU
- ▶ Cola de bloqueados por recurso: procesos a la espera de finalizar una petición bloqueante al recurso asociado
- Un proceso solo puede estar en una cola (como mucho)



# Multiprogramación (datos)

## Contexto de un proceso

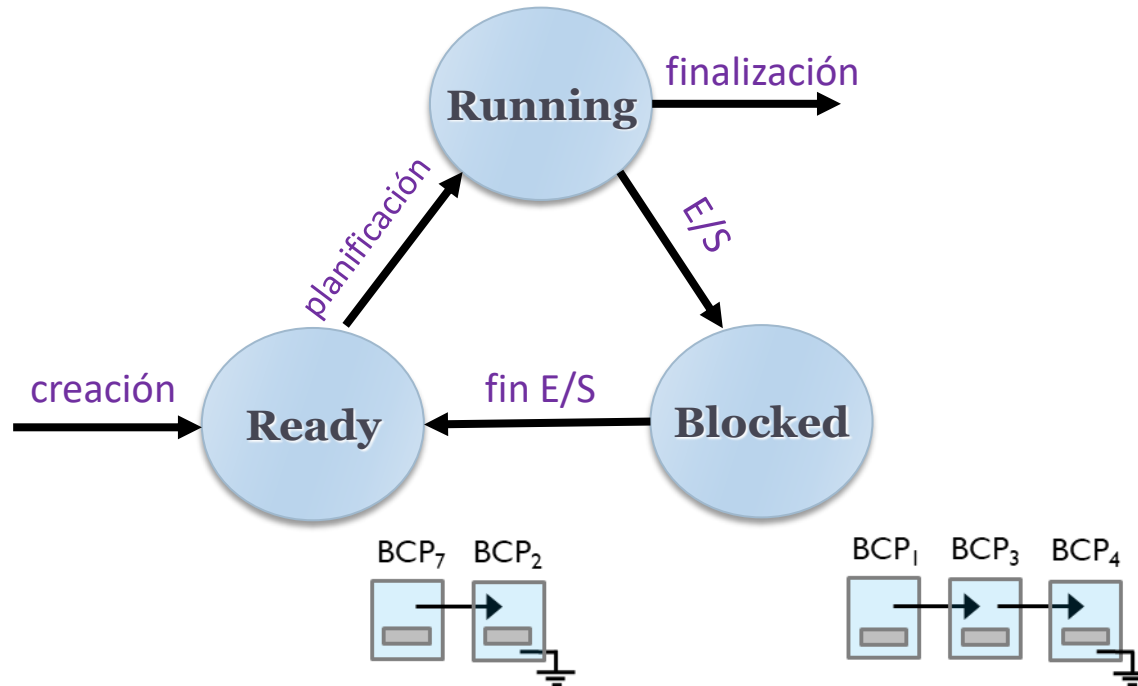
- Estado
- Lista/Cola
- Contexto



# Multiprogramación (datos)

## Contexto de un proceso

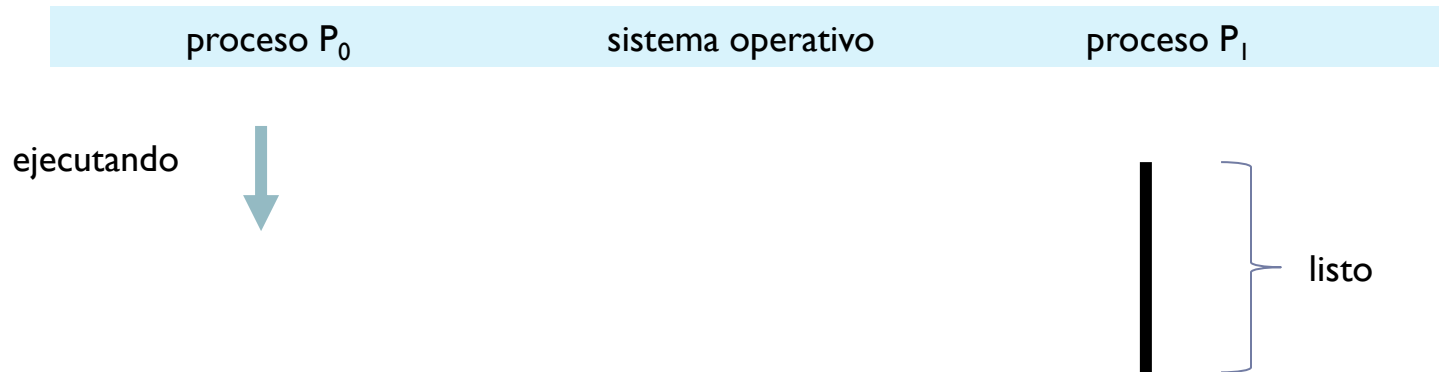
- Estado
- Lista/Cola
- Contexto



- ▶ **Registros generales:** PC, RE, etc.
- ▶ **Registros específicos:** Registros de coma flotante, etc.
- ▶ **Referencias a recursos:** puntero a código, datos, etc.

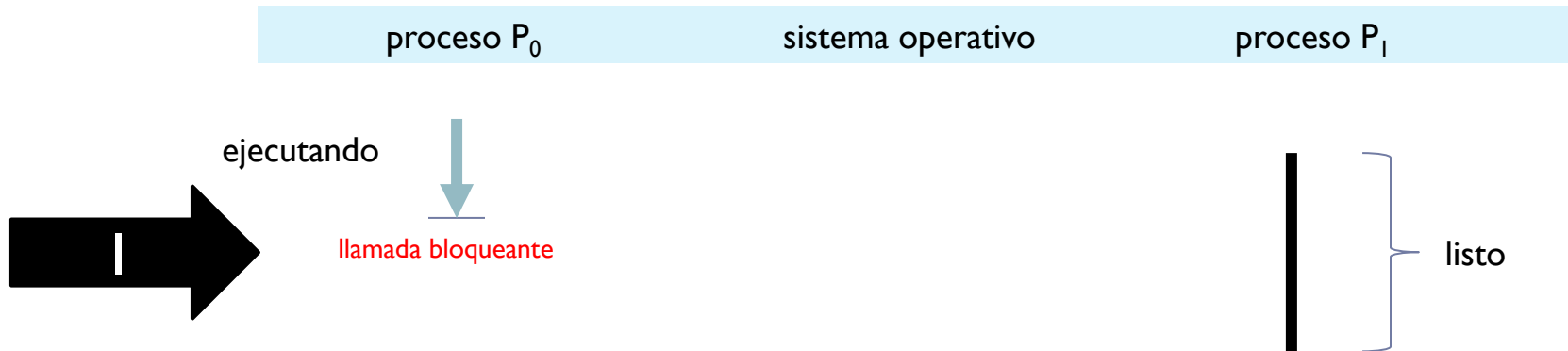
# Multiprogramación: cambios de contexto

---

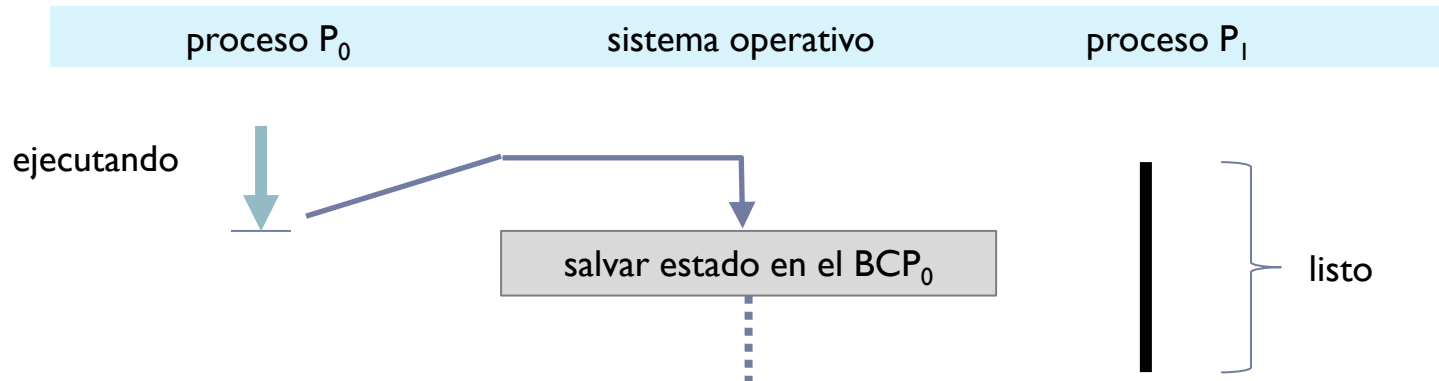


# Multiprogramación: cambios de contexto

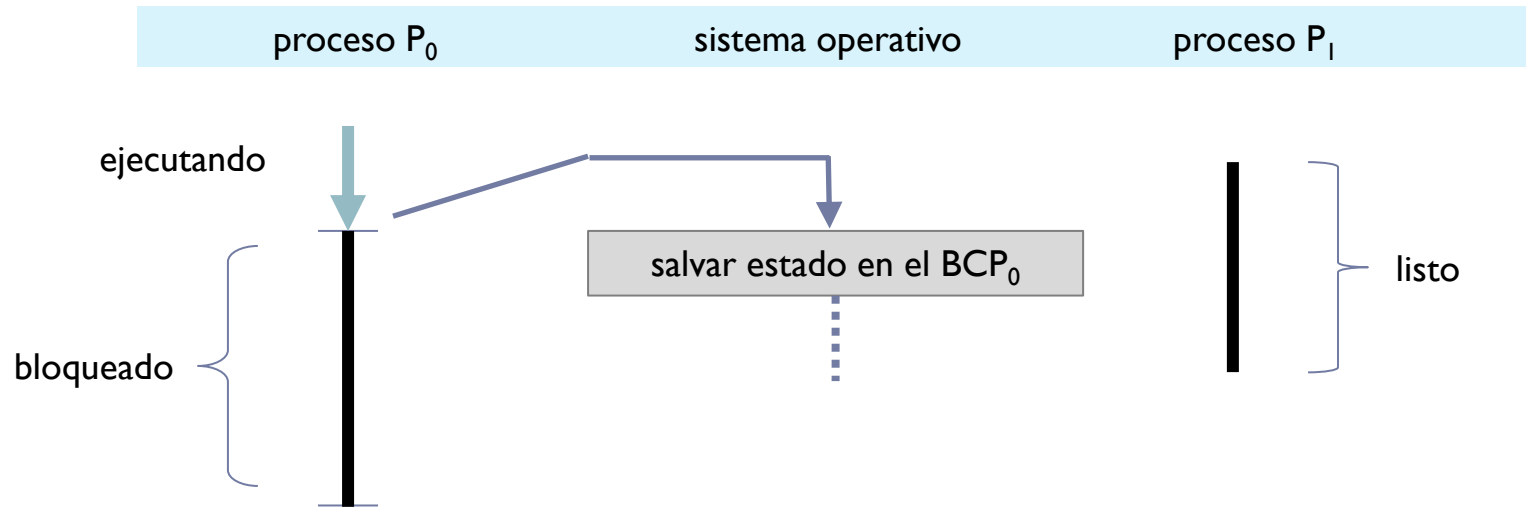
---



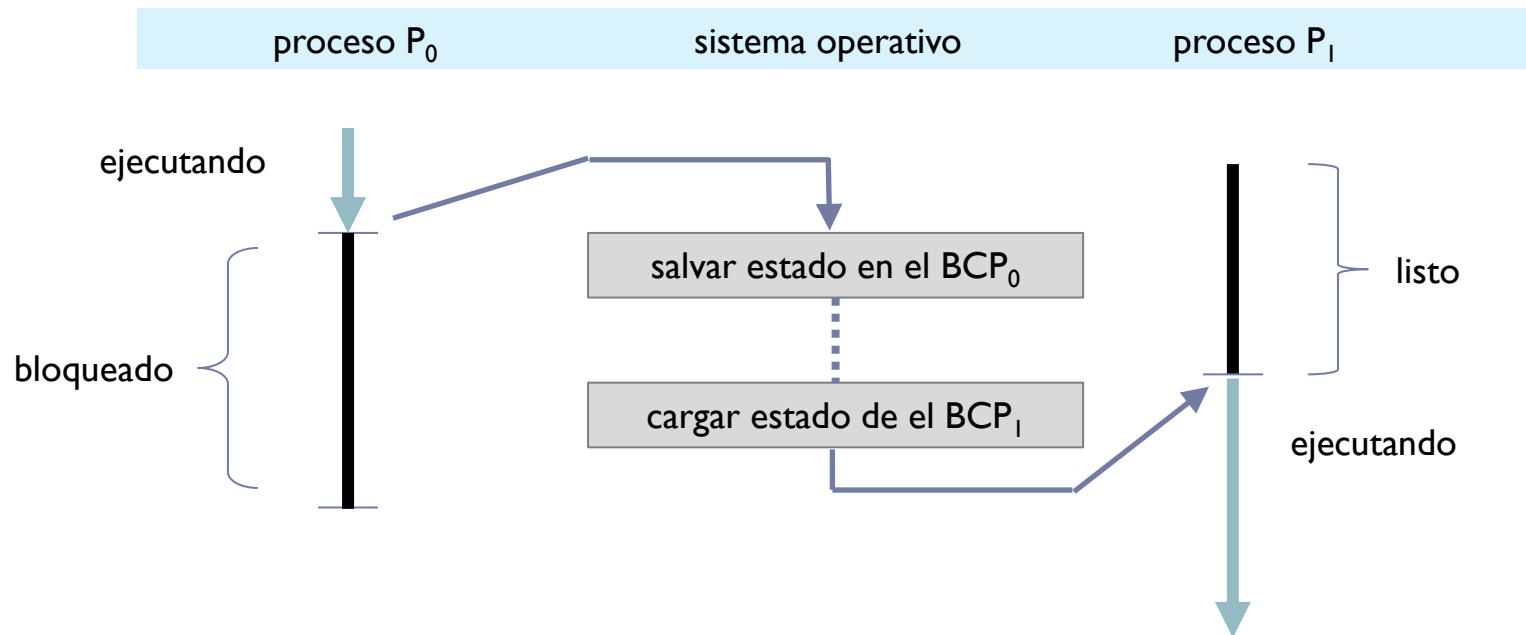
# Multiprogramación: cambios de contexto



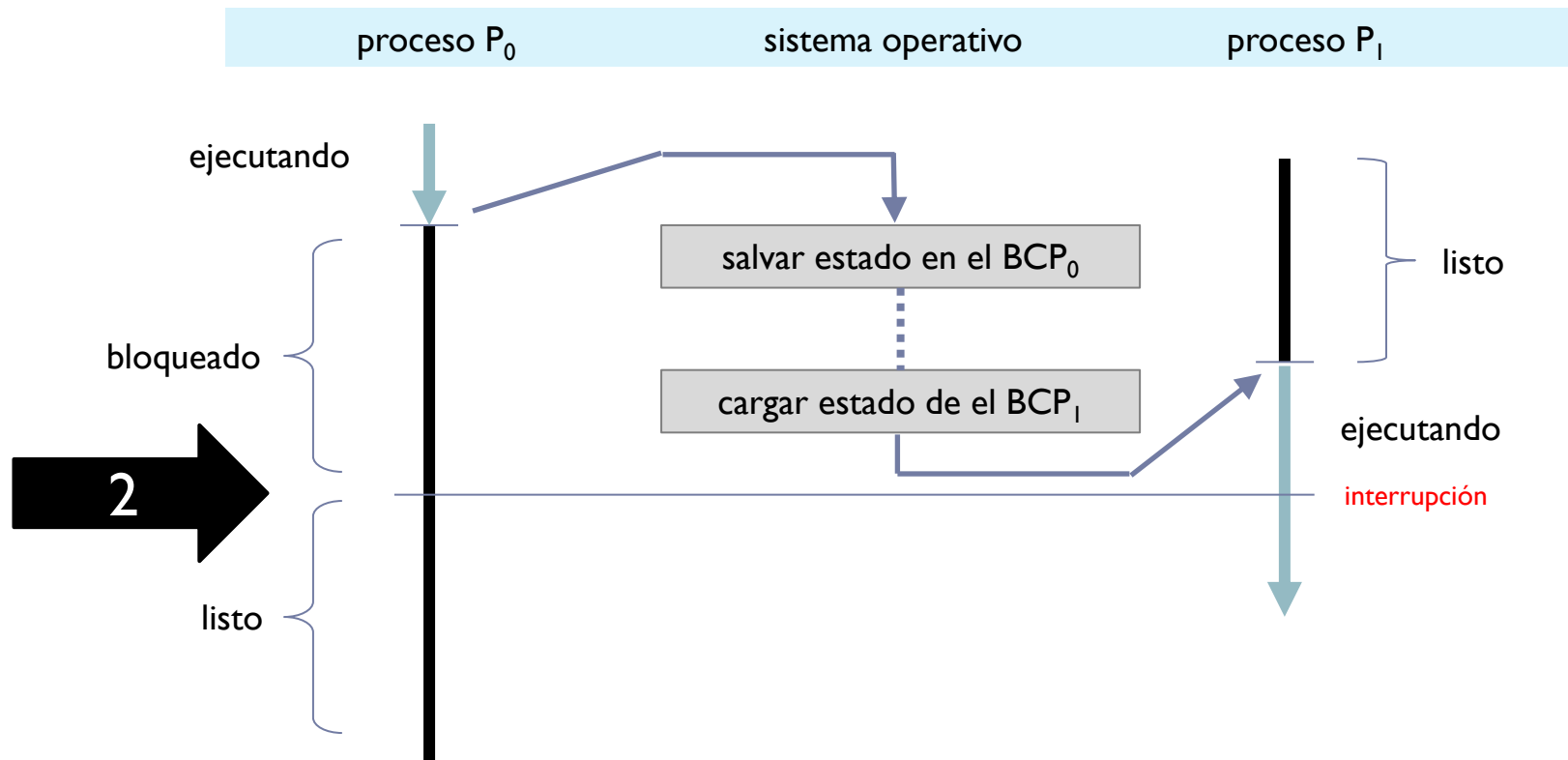
# Multiprogramación: cambios de contexto



# Multiprogramación: cambios de contexto

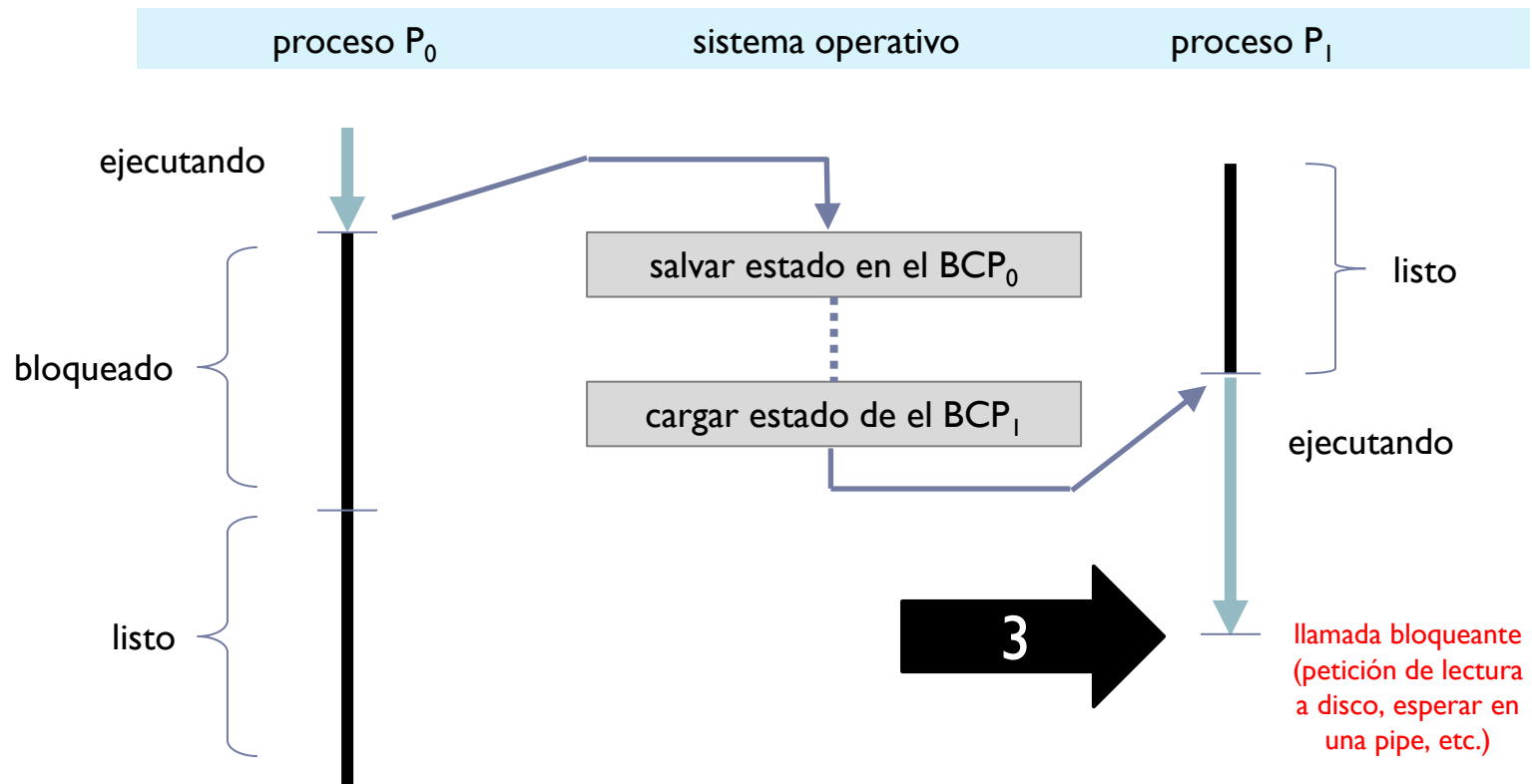


# Multiprogramación: cambios de contexto

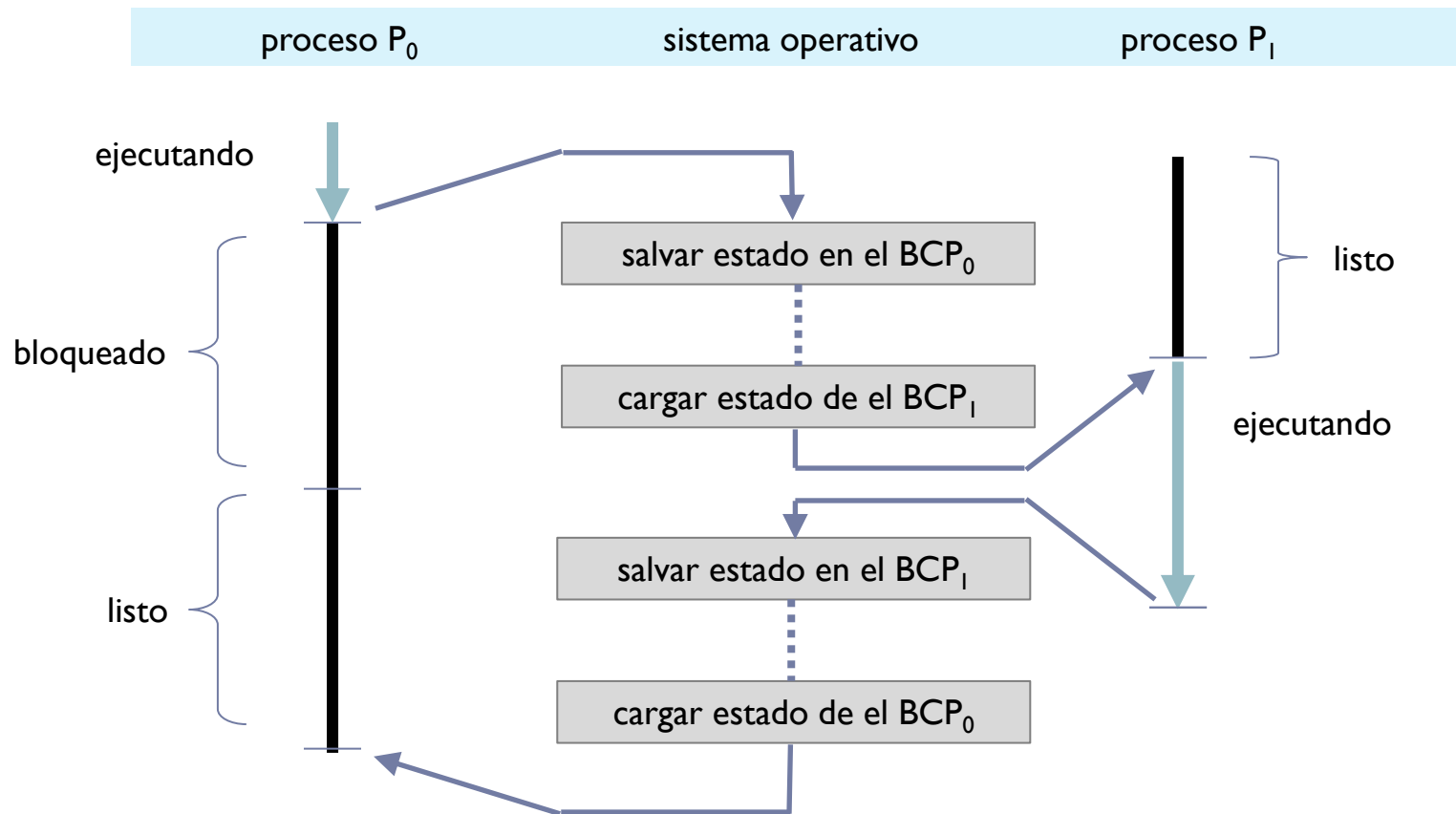




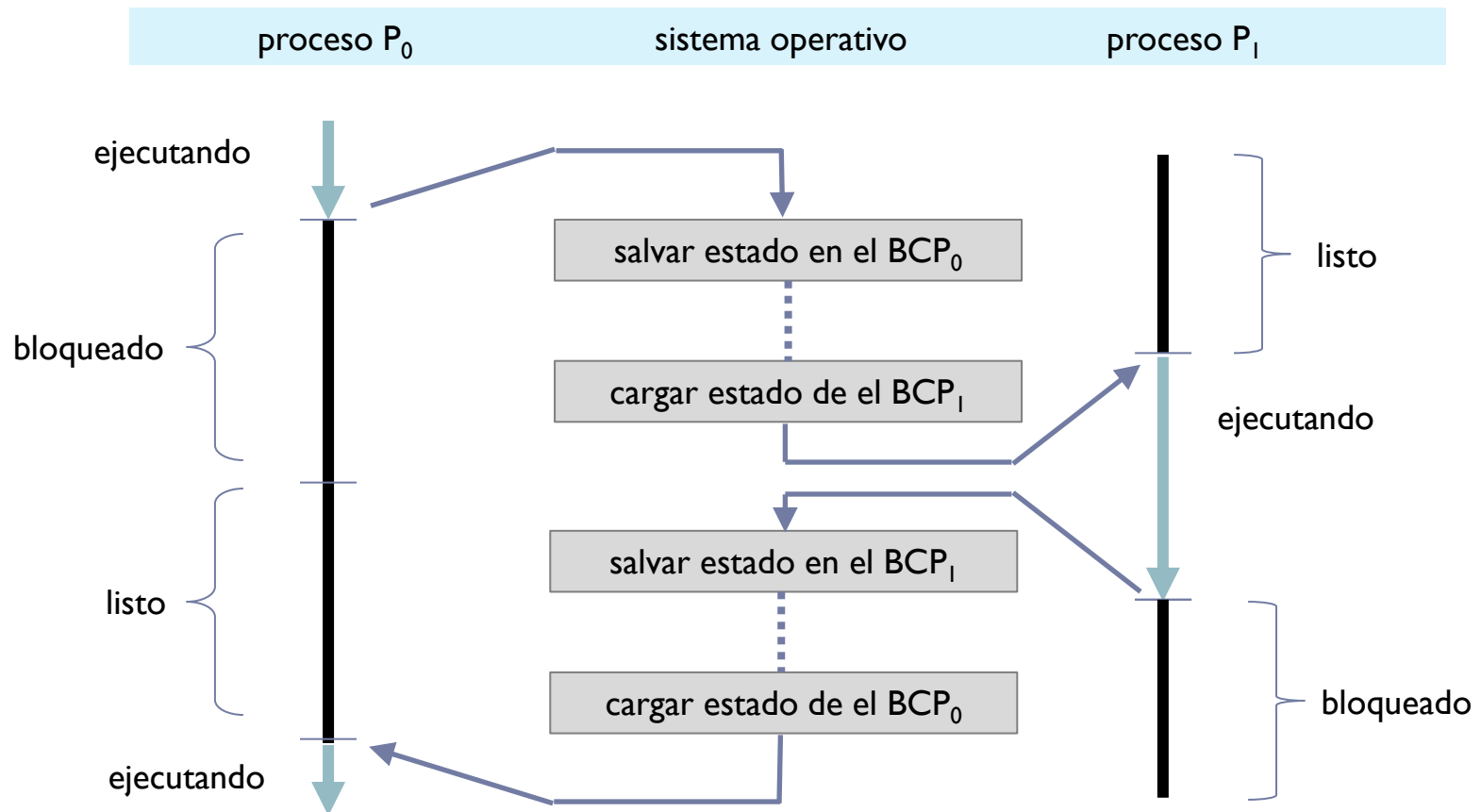
# Multiprogramación: cambios de contexto



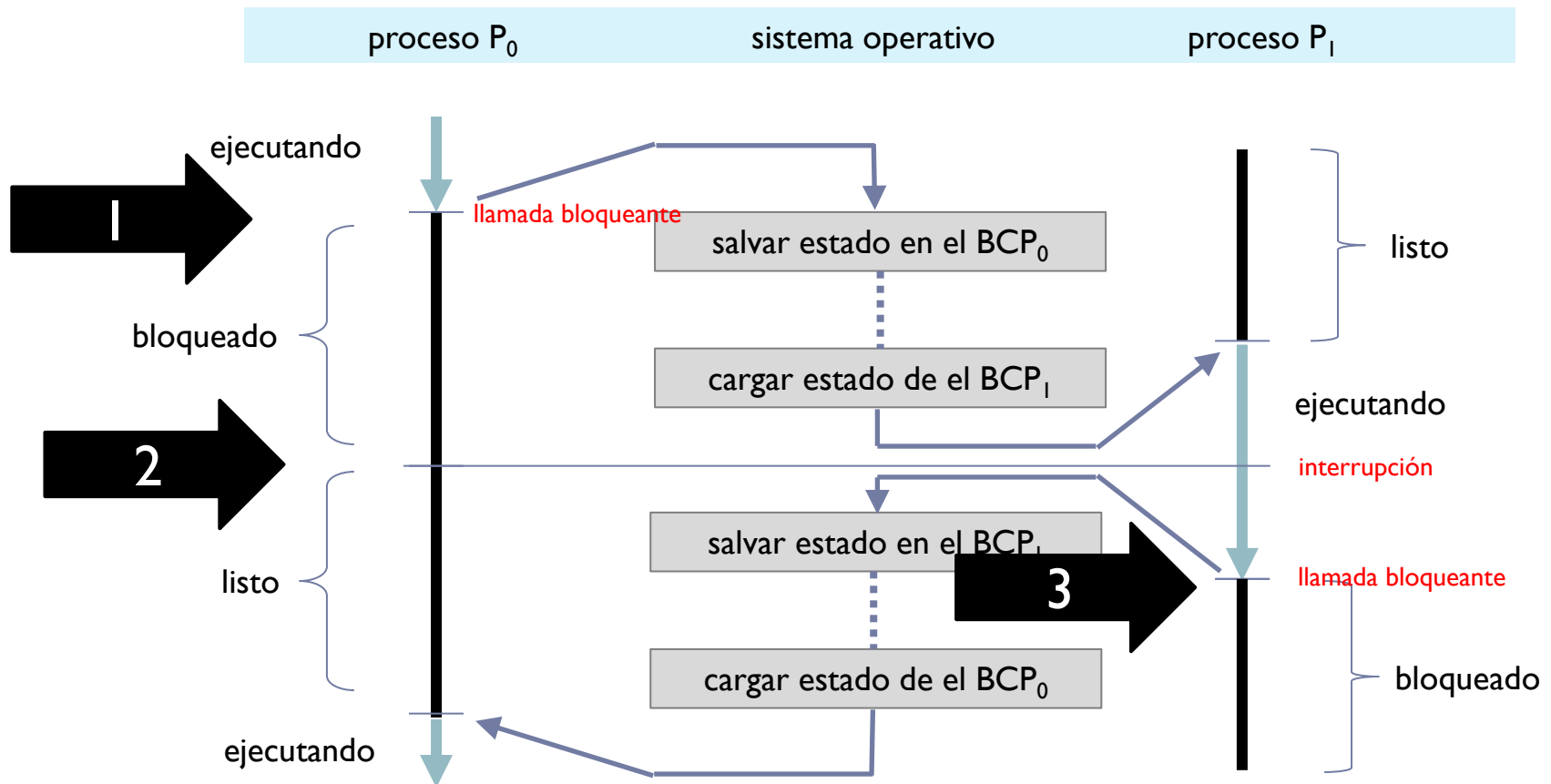
# Multiprogramación: cambios de contexto



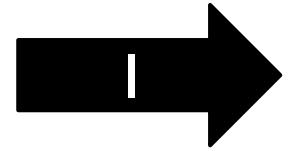
# Multiprogramación: cambios de contexto



# Multiprogramación: cambios de contexto



# Pseudocódigo de ejemplo (P0)



planificador()

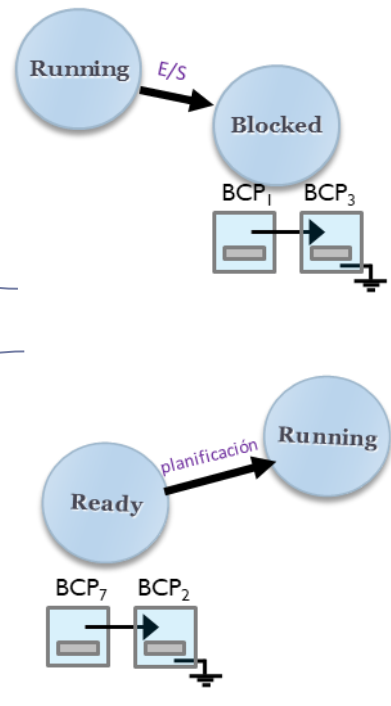
- return extraer(CPU\_Listos);

Teclado\_LeerTecla()

- Si (esVacio(Teclado\_Teclas))
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Teclado\_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = planificador();
- procesoActual->estado = EJECUCION;
- cambio\_contexto( &(proceso->contexto),  
&(procesoActual->contexto));
- return extraer(Teclado\_Teclas) ;

salvar estado en BCP<sub>0</sub>

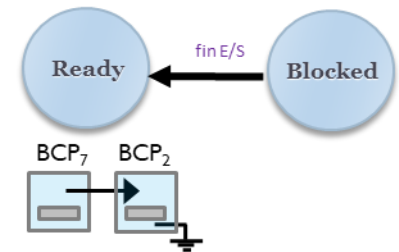
cargar estado en BCP<sub>1</sub>



# Pseudocódigo de ejemplo (P1)

Teclado\_Interrupción\_Hardware ()

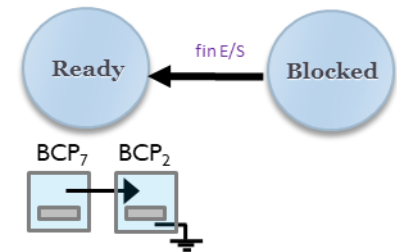
- T = in (TECLADO\_HW\_ID);
- proceso = **insertar** (T, Teclado\_Teclas);
- proceso = **primero** (Teclado\_Procesos);
- SI (proceso != NULL)
  - **eliminar** (Teclado\_Procesos);
  - proceso->estado = LISTO;
  - **insertar** (CPU\_Listos, proceso);
- return ok;



# Pseudocódigo de ejemplo (P1)

Teclado\_Interrupción\_Hardware ()

- T = in (TECLADO\_HW\_ID);
- proceso = **insertar** (T, Teclado\_Teclas);
- proceso = **primero** (Teclado\_Procesos);
- SI (proceso != NULL)
  - **eliminar** (Teclado\_Procesos);
  - proceso->estado = LISTO;
  - **insertar** (CPU\_Listos, proceso);
- return ok;



- Un proceso solo puede estar en una cola (como mucho):  
 [correcto] eliminar + insertar  
 [incorrecto] insertar + eliminar

# Pseudocódigo de ejemplo (P1)

planificador()

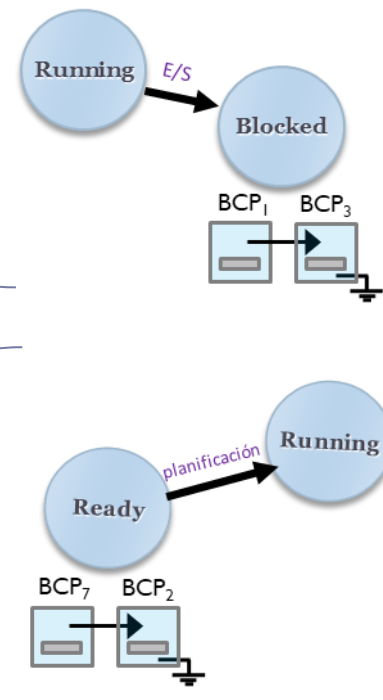
- return extraer(CPU\_Listos);

Disco\_LeerBloqueDisco()

- Si (no hay bloque en caché)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Disco\_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = planificador();
- procesoActual->estado = EJECUCION;
- **cambio\_contexto**( &(proceso->contexto), &(procesoActual->contexto));
- return extraer(Disco\_caché, bloque) ;

salvar estado en BCP<sub>1</sub>

cargar estado en BCP<sub>0</sub>





# Pseudocódigo de ejemplo (P0)

## Disco\_LeerBloqueDisco()

- Si (no hay bloque en caché)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Disco\_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = planificador();
- procesoActual->estado = EJECUCION;
- cambio\_contexto( &(proceso->contexto),  
                            &(procesoActual->contexto));
- return extraer(Disco\_caché, bloque) ;



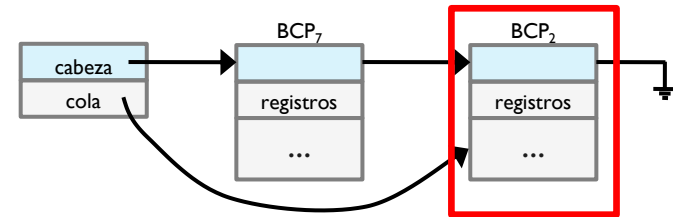
## Teclado\_LeerTecla()

- Si (no hay tecla)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Teclado\_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = planificador();
- procesoActual->estado = EJECUCION;
- cambio\_contexto( &(proceso->contexto),  
                            &(procesoActual->contexto));
- return extraer(Teclado\_Teclas) ;

# Planificador y activador

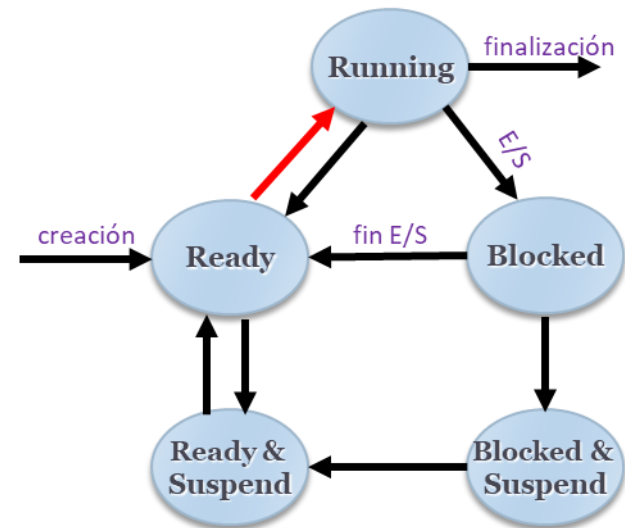
## ► Planificador:

Selecciona el proceso a ser ejecutado entre los que están listos para ejecutar



## ► Activador:

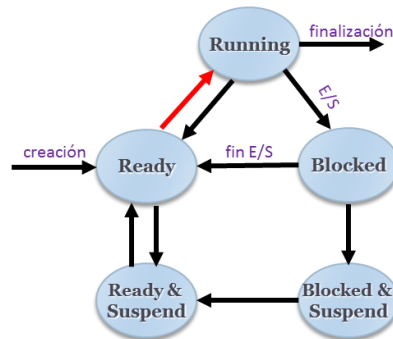
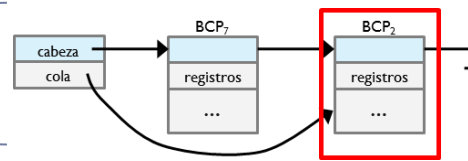
Da control al proceso que el planificador ha seleccionado (cambio de contexto - restaurar)



# Planificador y activador

## planificador()

- return extraer(CPU\_Listos);



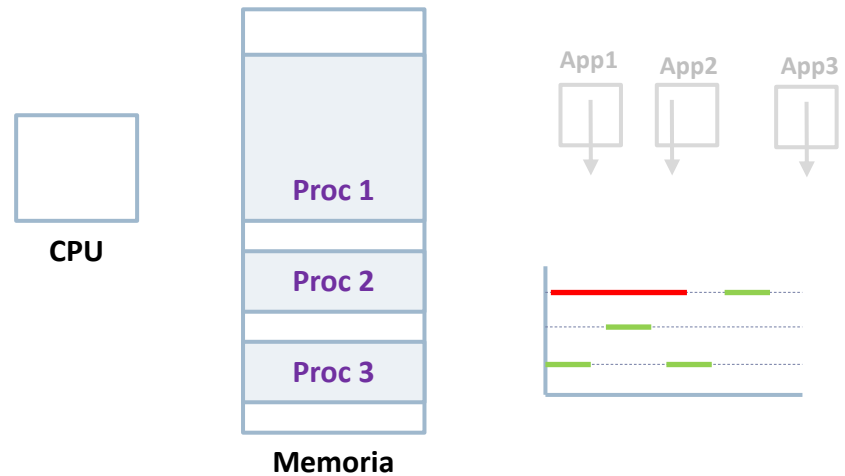
## Teclado\_LeerTecla()

- Si (no hay tecla)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Teclado\_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = **planificador()**;
- procesoActual->estado = EJECUCION;
- **activador** ( &(proceso->contexto),  
&(procesoActual->contexto));
- return extraer(Teclado\_Teclas) ;

# Modelo ofrecido

## repaso

- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- **multitarea**
- multiproceso



## ► Multitarea

- Cada proceso se ejecuta un quantum de tiempo (Ej.: 5 ms) y se rota el turno para ejecutar procesos no bloqueados
  - Cambio de contexto involuntario (C.C.I.)
- Reparto del uso del procesador
  - Parece que todo se ejecuta a la vez

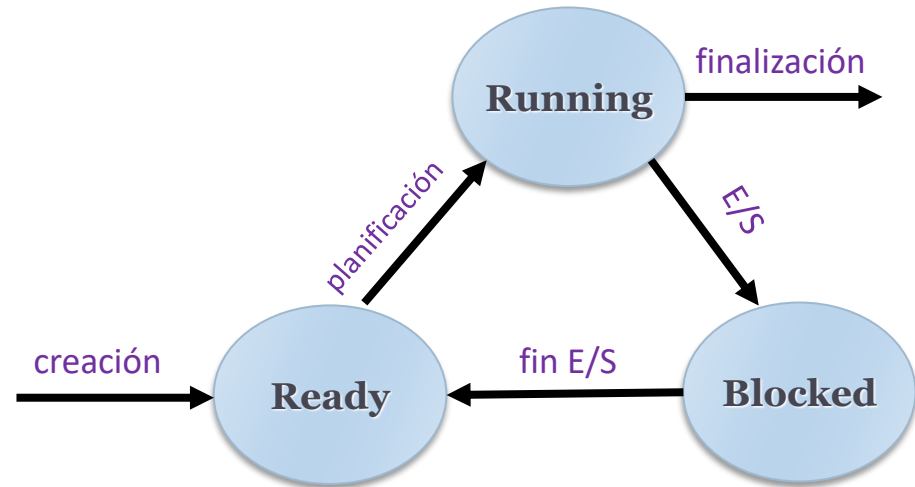
# Multitarea (datos y funciones)

Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	<ul style="list-style-type: none"> <li>Zonas de memoria (código, datos y pila)</li> <li>Archivos abiertos</li> <li>Señales activas</li> </ul>	<ul style="list-style-type: none"> <li>Diversas funciones internas</li> <li>Diversas funciones de servicio para memoria, ficheros, etc.</li> </ul>
Multiprogramación	<ul style="list-style-type: none"> <li>Estado de ejecución</li> <li>Contexto: registros de CPU...</li> <li>Lista de procesos</li> </ul>	<ul style="list-style-type: none"> <li>Int. hw/sw de dispositivos</li> <li>Planificador</li> <li>Crear/Destruir/Planificar proceso</li> </ul>
○ Protección / Compartición	<ul style="list-style-type: none"> <li>Paso de mensajes                             <ul style="list-style-type: none"> <li>Cola de mensajes de recepción</li> </ul> </li> <li>Memoria compartida                             <ul style="list-style-type: none"> <li>Zonas, locks y conditions</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Envío/Recepción mensaje y gestión de la cola de mensaje</li> <li>API concurrencia y gestión de estructuras de datos</li> </ul>
○ Jerarquía de procesos	<ul style="list-style-type: none"> <li>Relación de parentesco</li> <li>Conjuntos de procesos relacionados</li> <li>Procesos de una misma sesión</li> </ul>	<ul style="list-style-type: none"> <li>Clonar/Cambiar imagen de proceso</li> <li>Asociar procesos e indicar proceso representante</li> </ul>
Multitarea	<ul style="list-style-type: none"> <li>Quantum restante</li> <li>Prioridad</li> </ul>	<ul style="list-style-type: none"> <li>Int. hw/sw de reloj</li> <li>Planificador</li> <li>Crear/Destruir/Planificar proceso</li> </ul>
Multiproceso	<ul style="list-style-type: none"> <li>Afinidad</li> </ul>	<ul style="list-style-type: none"> <li>Int. hw/sw de reloj</li> <li>Planificador</li> <li>Crear/Destruir/Planificar proceso</li> </ul>

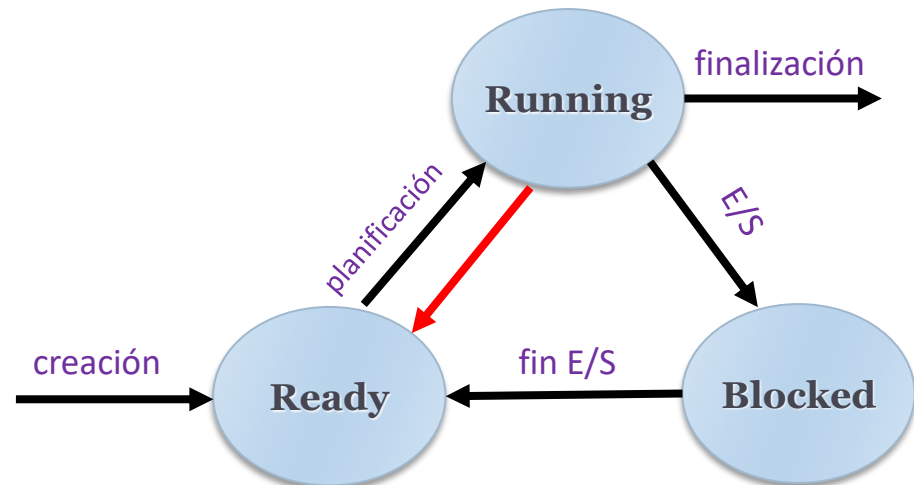
# Estados de un proceso

- Estado
- Lista/Cola
- Contexto

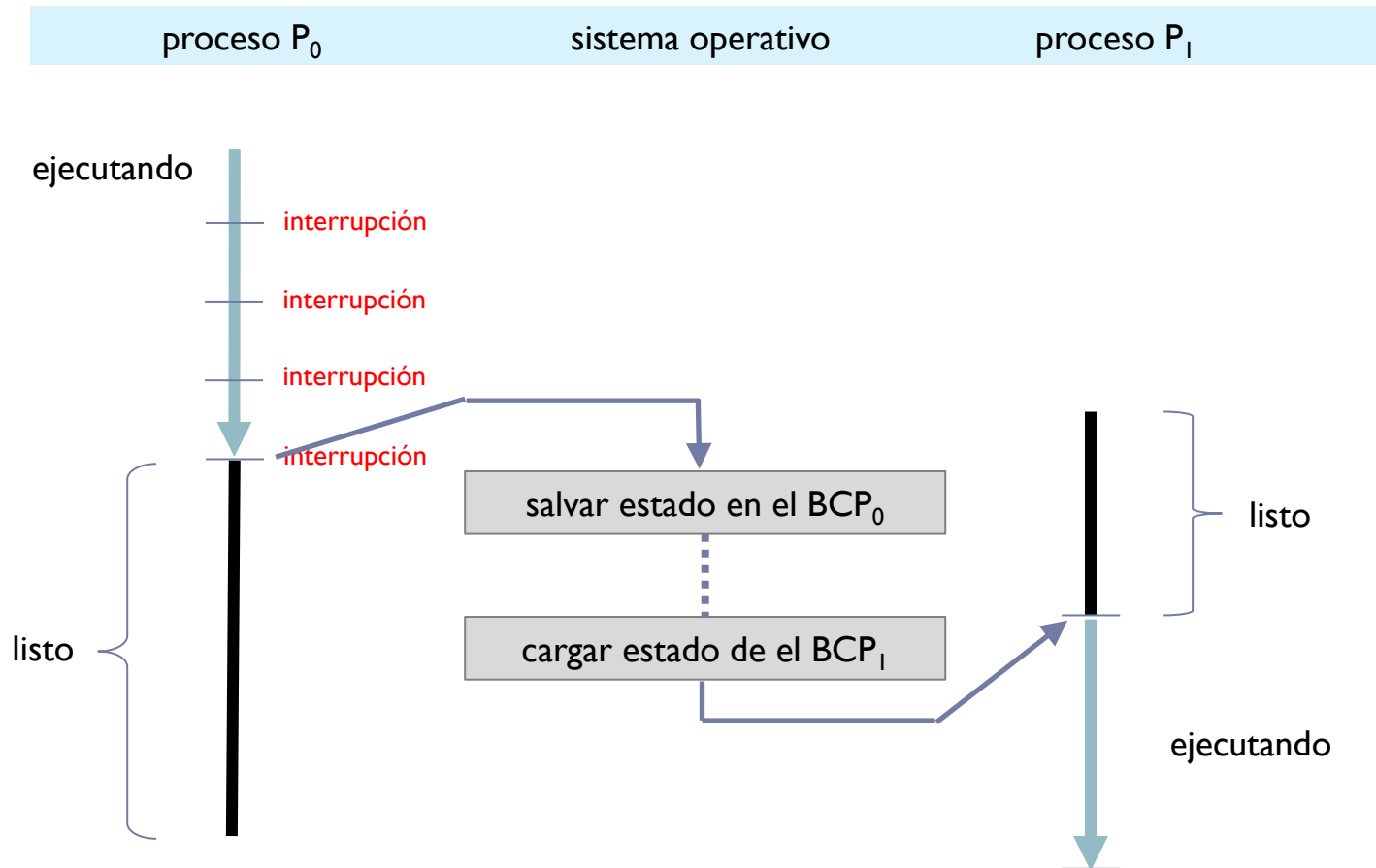
C.C.V.



C.C.V. + C.C.I.



# El reloj: tratamiento con c.c.v. + c.c.i.



# Pseudocódigo de ejemplo (P0)

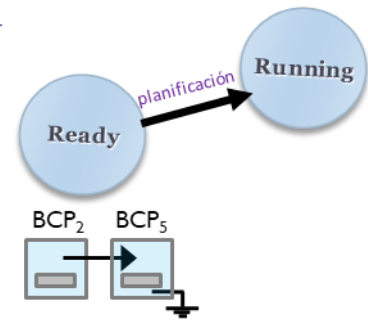
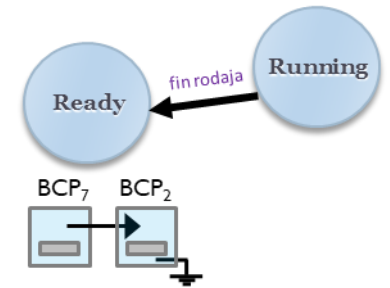
Reloj\_Interrupción\_Hardware ()

- Ticks++;
- pActual->rodaja = pActual->rodaja - 1;
- SI (pActual->rodaja == 0)
  - pActual->estado = LISTO;
  - pActual->rodaja = RODAJA;
  - **insertar** (CPU\_Listos, pActual);
  - proceso = pActual;
- pActual = planificador();
- pActual->estado = EJECUCIÓN;
- **cambio\_contexto**(  
    &(proceso->contexto),  
    &(pActual->contexto));
- return ok;

planificador()  
• return extraer(CPU\_Listos);

salvar estado en BCP<sub>0</sub>

cargar estado en BCP<sub>1</sub>





# Tipos de cambio de contexto

## resumen

---

### ▶ 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.
- ▶ **Transición:** *En ejecución → bloqueado.*
- ▶ **Escenarios:** leer del terminal, fallo de página, etc.
- ▶ **Motivo:** *Eficiencia en el uso del procesador*

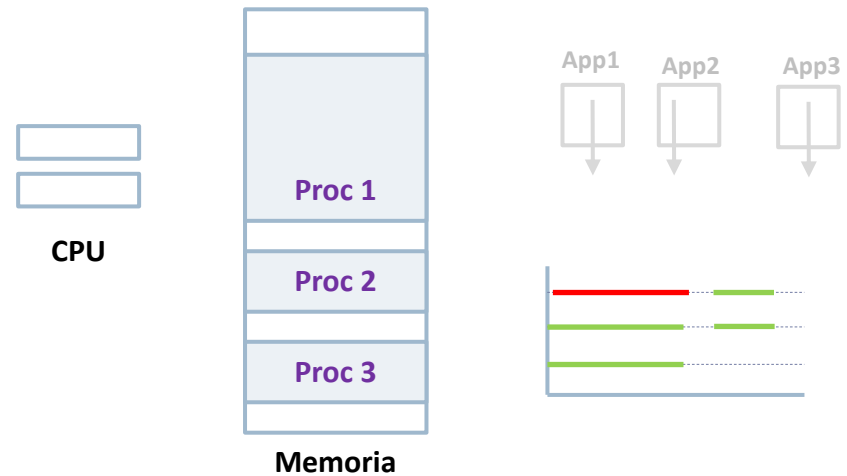
### ▶ Cambio de contexto *involuntario* (C.C.I):

- ▶ SO quita de la CPU al proceso
- ▶ **Transición:** *En ejecución → listo*
- ▶ **Escenarios:** fin de rodaja u otro proceso de mayor prioridad pasa a *listo*
- ▶ **Motivo:** *Reparto del uso del procesador*

# Modelo ofrecido

## repaso

- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- multitarea
- **multiproceso**

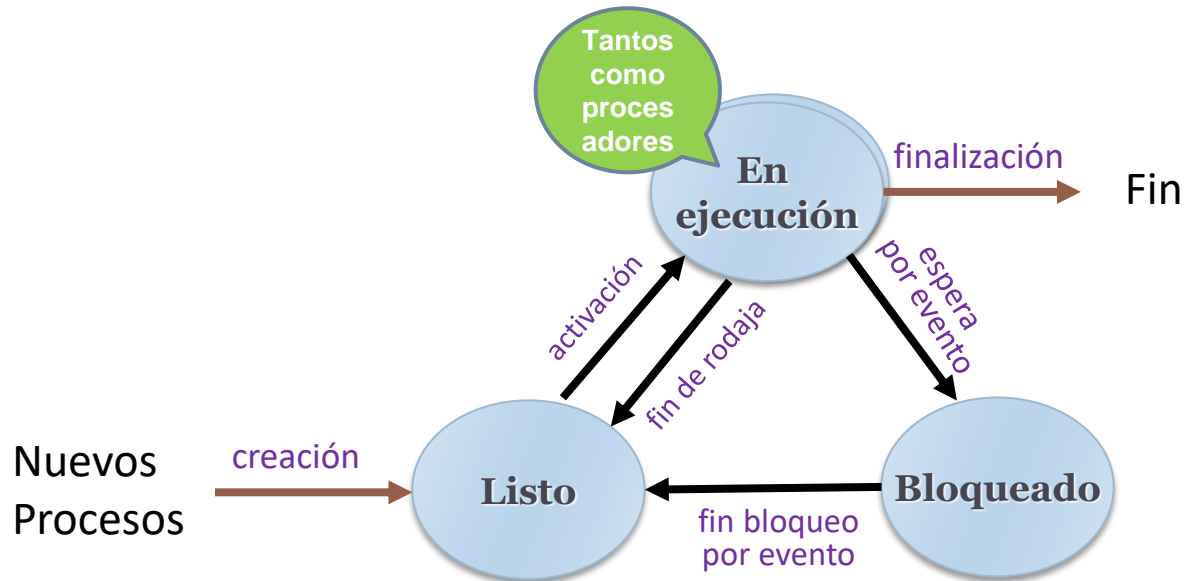


## ► Multiproceso

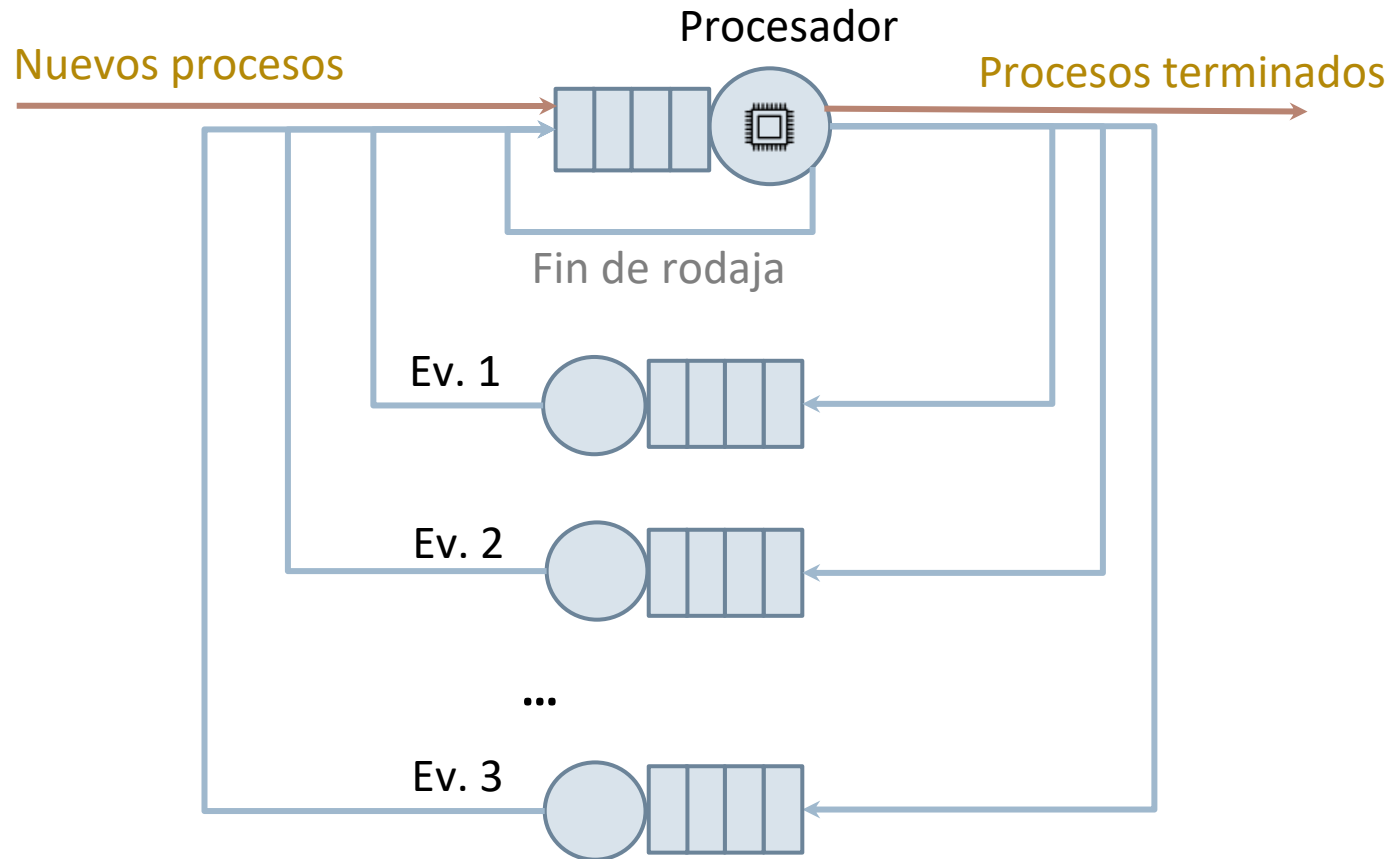
- Se dispone de varios procesadores (multicore/multiprocesador)
- Además del reparto de cada CPU (multitarea) hay paralelismo real entre varias tareas (tantas como procesadores)
  - Se suele usar planificador y estructuras de datos separadas por procesador con algún mecanismo de equilibrio de carga

# Ciclo de vida básico de un proceso

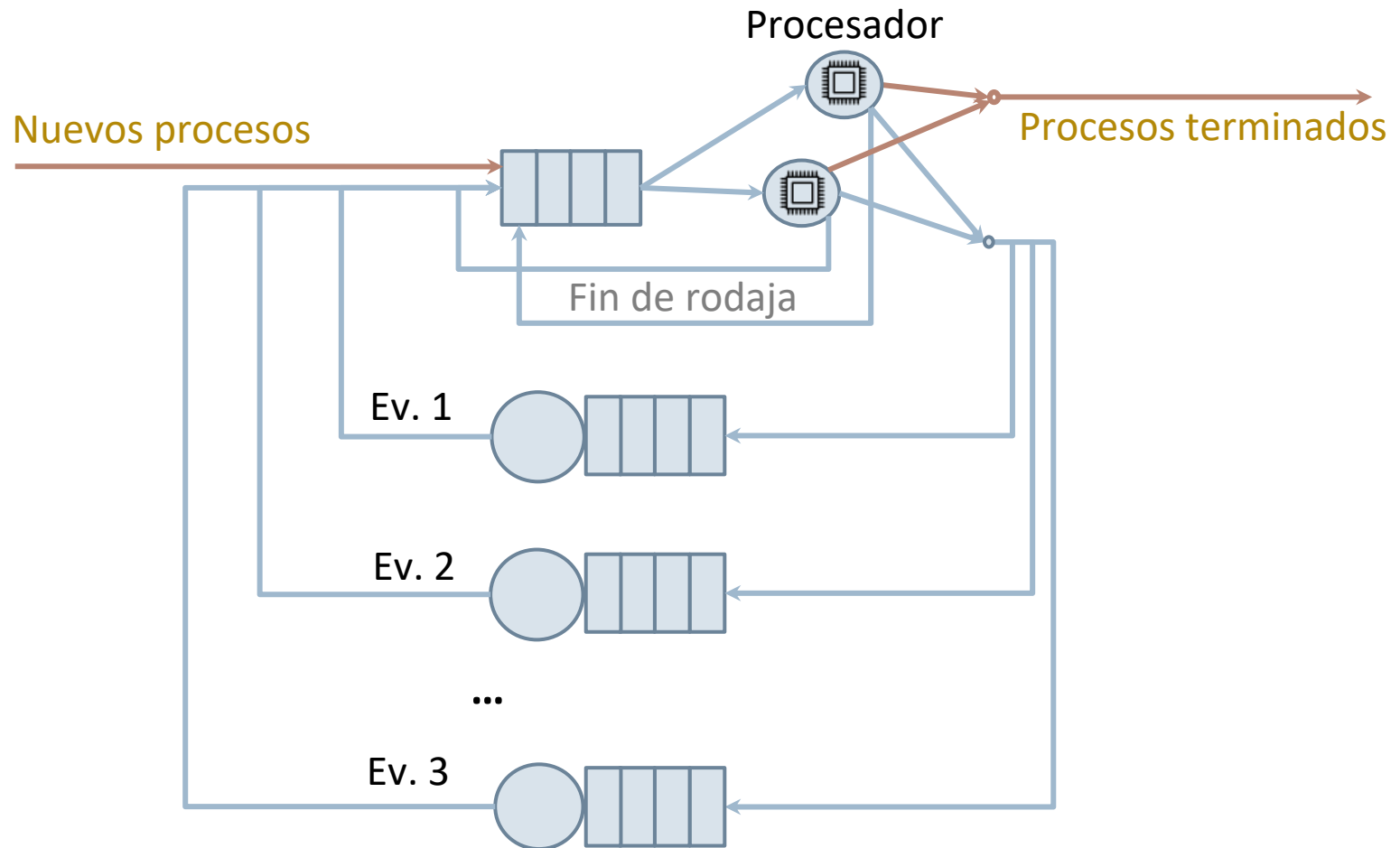
- Estado
- Lista/Cola
- Contexto



# Modelo de colas simplificado: 1 procesador

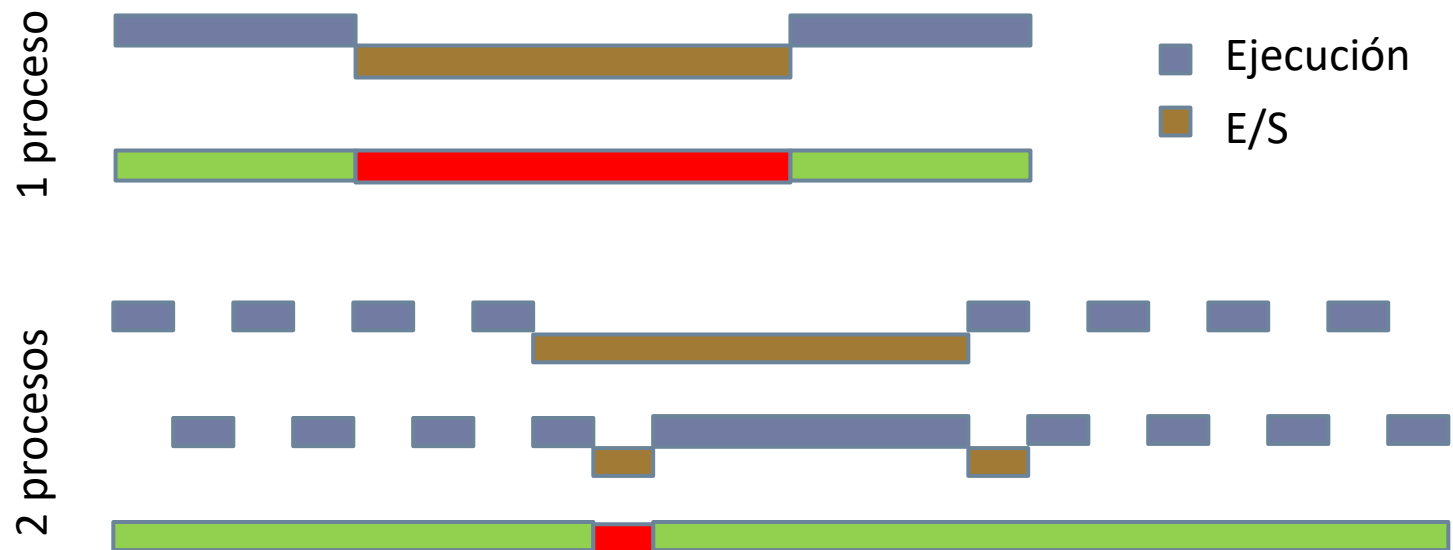


# Modelo de colas simplificado: N procesadores



# Ventajas de la multitarea

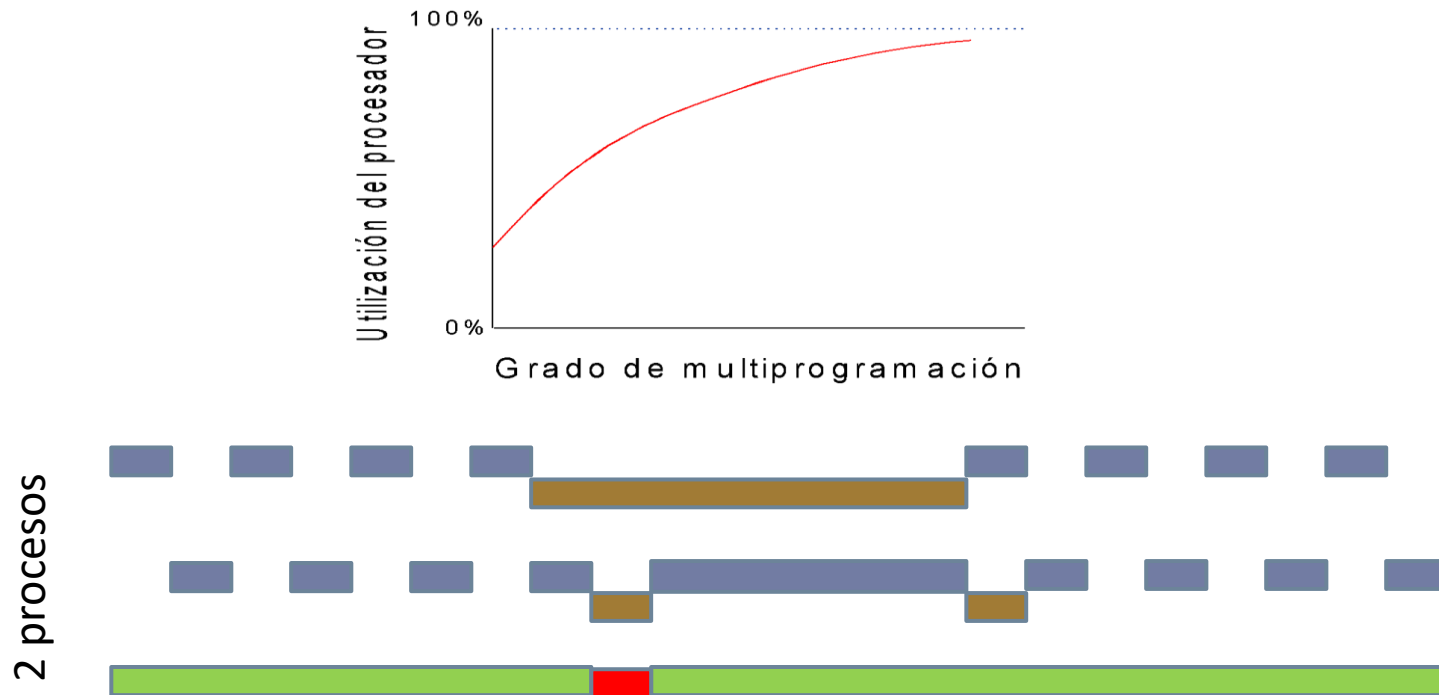
- ▶ Modularidad: facilita la programación dividiendo los programas en procesos.
- ▶ Permite el servicio interactivo simultáneo de N usuarios de forma eficiente.
- ▶ Aprovecha los tiempos que los procesos pasan esperando a que se completen sus operaciones de E/S.
- ▶ Aumenta el uso de la CPU.



# Ventajas de la multitarea

Proceso A
Proceso B
Proceso C
SO
Memoria principal

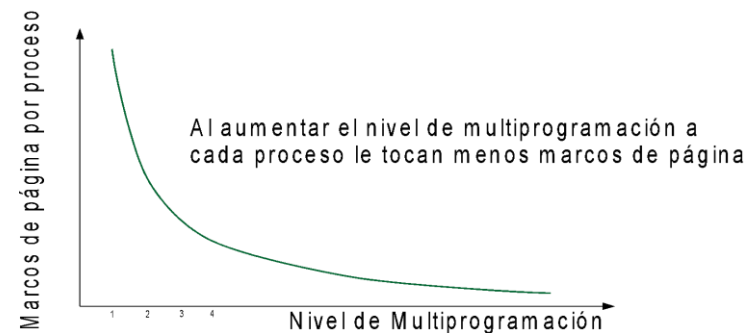
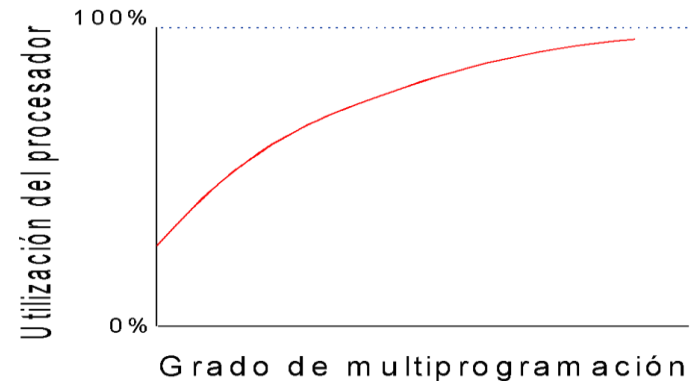
- ▶ El uso de la CPU... depende del grado de multiprogramación
- ▶ Grado de multiprogramación: nº de procesos activos.
- ▶ ¿Siempre más procesos mejora el % de utilización de la CPU?



# Multiprogramación y memoria

Proceso A
Proceso B
Proceso C
SO
Memoria principal

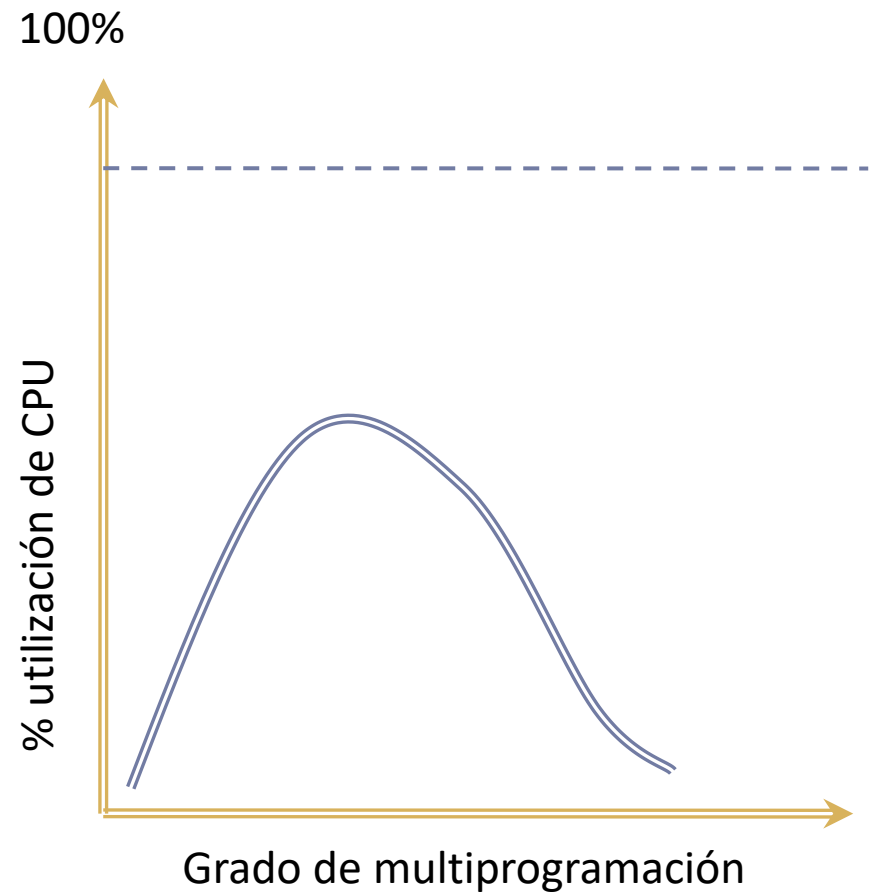
- ▶ Los sistemas **sin** memoria virtual:
  - ▶ Cada proceso reside totalmente en M.P.
- ▶ Los sistemas **con** memoria virtual:
  - ▶ Dividen el espacio direccionable de los procesos en páginas.
  - ▶ Dividen la memoria física principal en marcos de página.
  - ▶ En un momento dado cada proceso tiene un cierto número de sus páginas en memoria principal (**conjunto residente**).





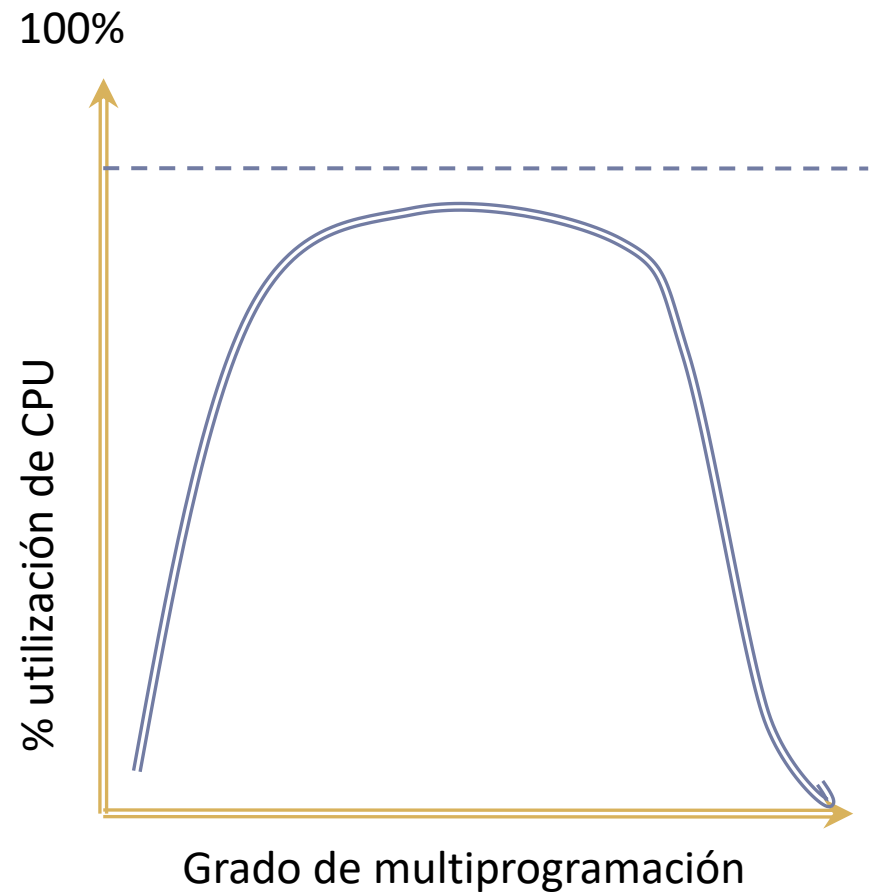
# Rendimiento: Poca memoria física

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



# Rendimiento: Mucha memoria física

- ▶ Al aumentar el grado de multiprogramación:
  - ▶ Desciende el tamaño del conjunto residente de cada proceso.
- ▶ Mucha memoria: se alcanza un alto % de utilización de CPU con menos procesos de los que caben en memoria.
- ▶ **Problema:** memoria “de más”.  
**Solución:** Mejora del procesador o incorporación de más procesadores.



# Ciclo de vida básico de un proceso

- Estado
- Lista/Cola
- Contexto



- El SO puede expulsar totalmente procesos al swap si % uso CPU baja por hiperpaginación.
- Precisa de nuevos estados: bloqueado y suspendido + listo y suspendido.

# Contenidos

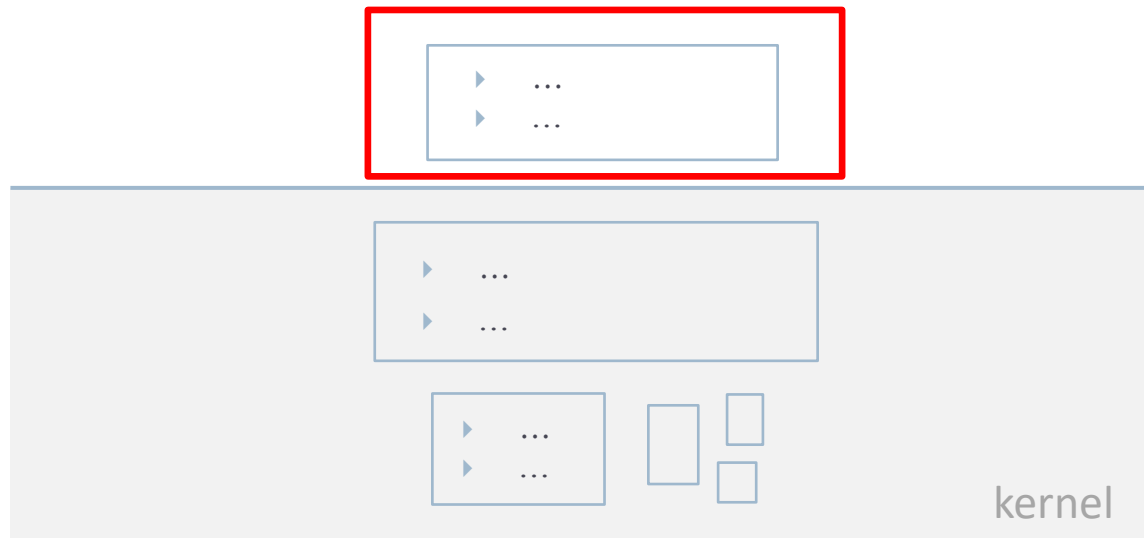
---

1. Introducción
  - Definición de proceso.
  - Modelo ofrecido: recursos, multiprogramación, multitarea y multiproceso
2. Ciclo de vida del proceso: estado de procesos.
3. **Servicios para gestionar procesos que da el sistema operativo.**
4. Definición de hilo o *thread*
5. Hilos de biblioteca y núcleo.
6. Servicios para hilos en el sistema operativo.

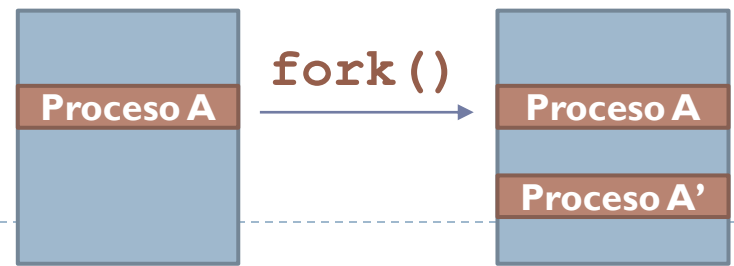
# Servicios del sistema operativo

## servicios POSIX de gestión de procesos

---

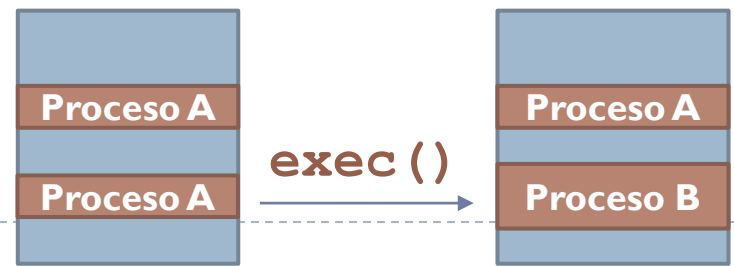


# Servicio fork



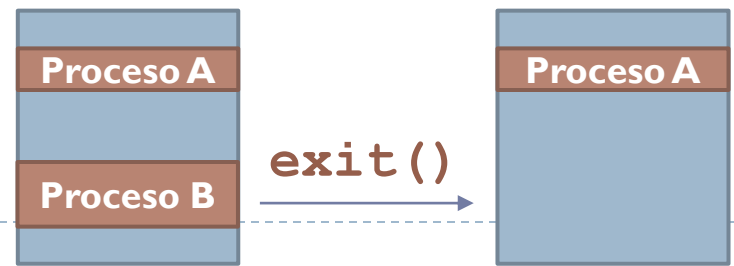
Servicio	<pre>#include &lt;unistd.h&gt;  pid_t fork(void);</pre>
Argumentos	
Devuelve	<ul style="list-style-type: none"><li>❑ -1 el caso de error.</li><li>❑ En el proceso padre: el identificador del proceso hijo.</li><li>❑ En el proceso hijo: 0</li></ul>
Descripción	<ul style="list-style-type: none"><li>❑ Duplica el proceso que invoca la llamada.</li><li>❑ Los procesos padre e hijo siguen ejecutando el mismo programa.</li><li>❑ El proceso hijo hereda los ficheros abiertos del proceso padre.<ul style="list-style-type: none"><li>❑ Se copian los descriptores de archivos abiertos.</li></ul></li><li>❑ Se desactivan las alarmas pendientes.</li></ul>

# Servicio exec



Servicio	<pre>#include &lt;unistd.h&gt; int <b>execl</b>(const char *path, const char *arg, ...); int <b>execv</b>(const char* path, char* const argv[]); int <b>execve</b>(const char* path, char* const argv[], char* const envp[]); int <b>execvp</b>(const char *file, char *const argv[]);</pre>
Argumentos	<ul style="list-style-type: none"><li>□ <code>path</code>: Ruta al archivo ejecutable.</li><li>□ <code>file</code>: Busca el archivo ejecutable en todos los directorios especificados por <code>PATH</code></li></ul>
Devuelve	<ul style="list-style-type: none"><li>□ Devuelve <code>-1</code> en caso de error, <b>en caso contrario no retorna.</b></li></ul>
Descripción	<ul style="list-style-type: none"><li>□ Cambia la imagen del proceso actual.</li><li>□ El mismo proceso ejecuta otro programa.</li><li>□ Los ficheros abiertos permanecen abiertos.</li><li>□ Las señales con la acción por defecto seguirán por defecto, las señales con manejador tomarán la acción por defecto</li></ul>

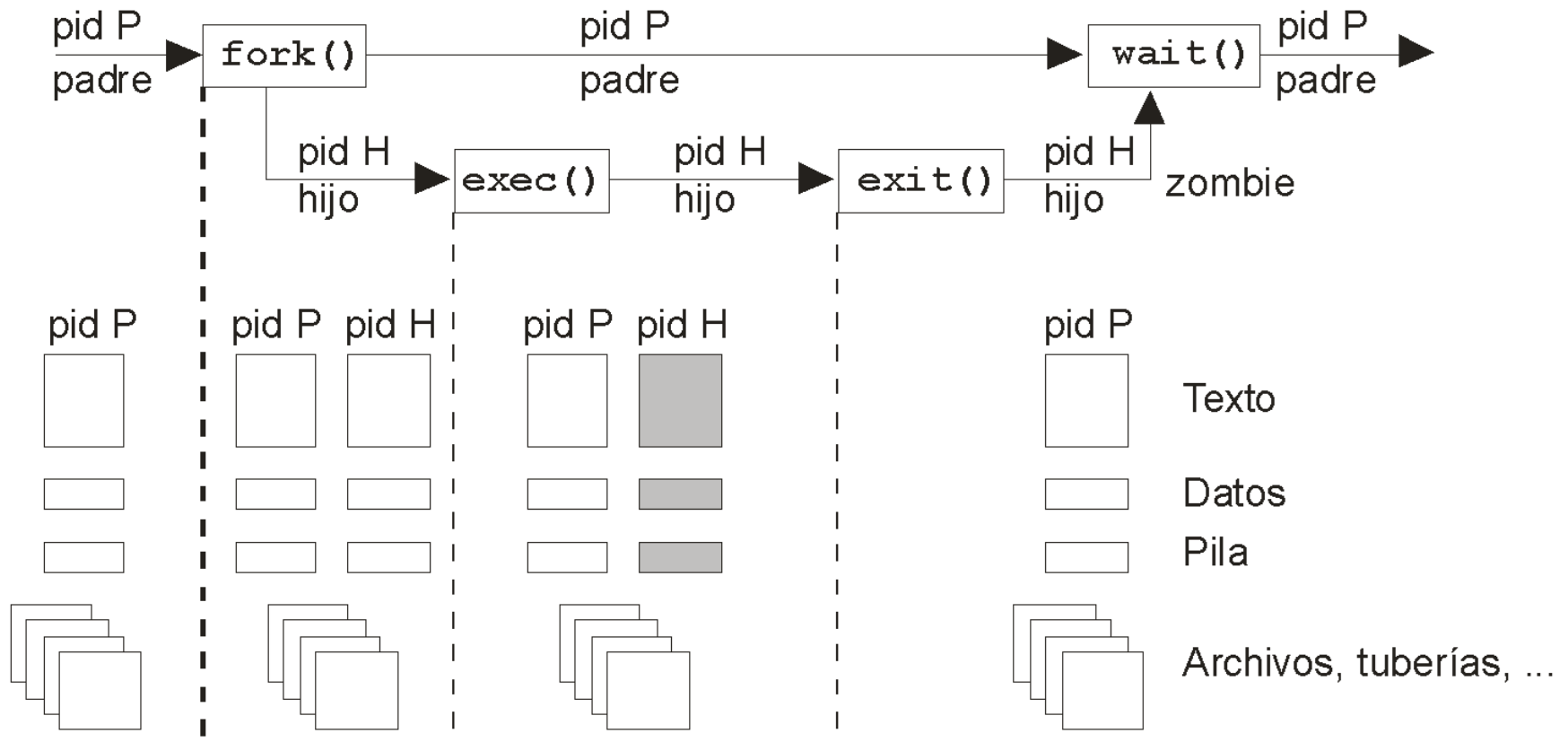
# Servicio exit



Servicio	<pre>#include &lt;unistd.h&gt;  void <b>exit</b>(status);</pre>
Argumentos	<ul style="list-style-type: none"><li>□ status: valor que el padre recupera en la llamada wait()</li></ul>
Devuelve	
Descripción	<ul style="list-style-type: none"><li>□ Finaliza la ejecución del proceso.</li><li>□ Se cierran todos los descriptores de ficheros abiertos.</li><li>□ Se liberan todos los recursos del proceso.</li><li>□ Se libera el BCP del proceso.</li></ul>



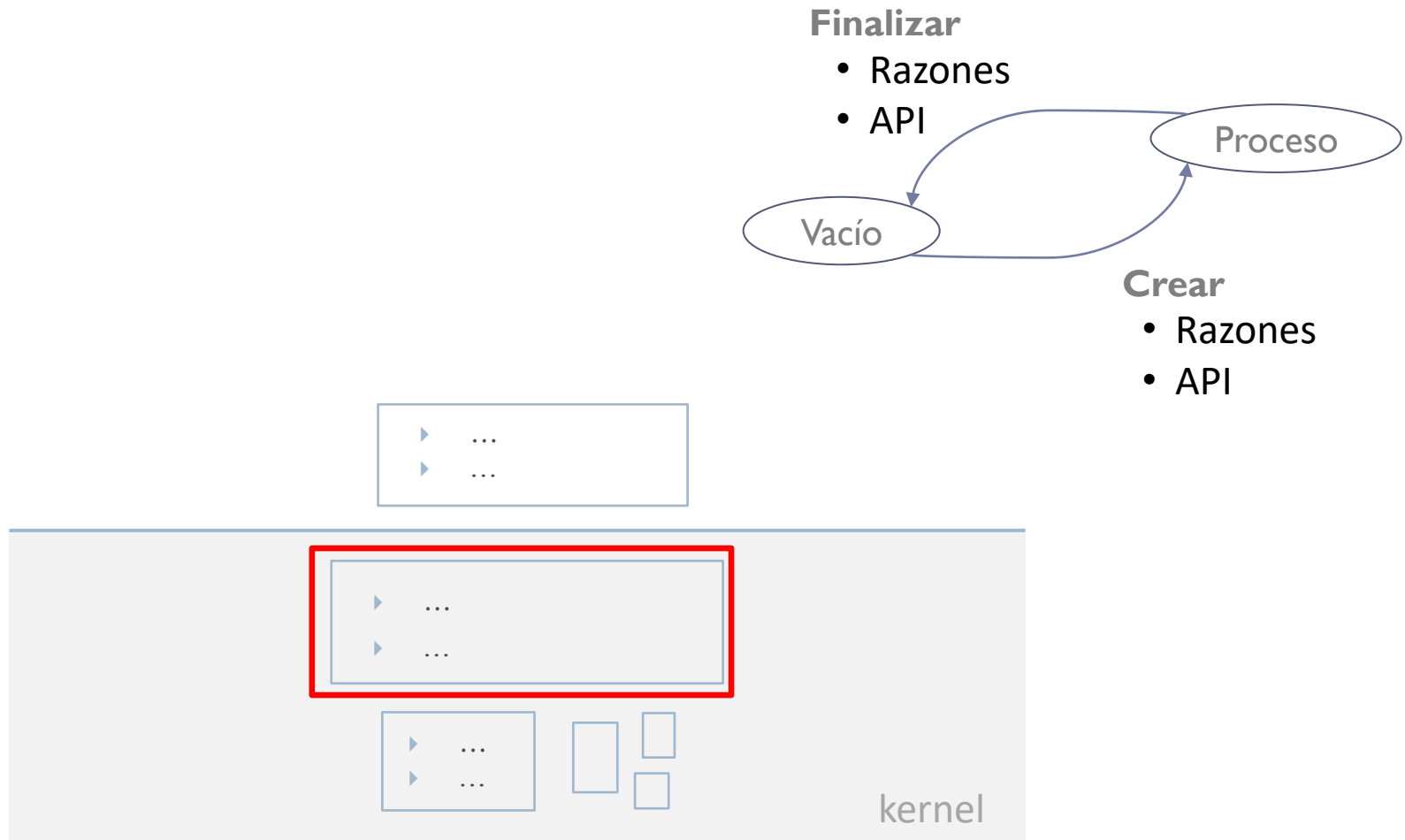
# Uso de los servicios fork, exec, wait y exit



# Servicios del sistema operativo

## inicialización y finalización de procesos

---



# Creación de procesos

---

## ▶ Un proceso se crea:

- ▶ Durante el arranque del sistema:
  - ▶ Hilos del kernel + primer proceso (Ej.: init, swapper, etc.)
- ▶ Cuando un proceso existe hace una llamada al sistema para crear otro:
  - ▶ Cuando el sistema operativo comienza un nuevo trabajo
  - ▶ Cuando un usuario arranca un nuevo programa
  - ▶ Cuando durante la ejecución de un programa se necesite

## ▶ Recursos:

- ▶ Los obtiene del SO, padre reparte recursos (parte o todo para evitar denegación de servicio por multiplicación). Ej.: **Copy-on-Write**
- ▶ El padre puede ejecutar en paralelo o esperar a la terminación de hijo.

# Finalización de procesos

---

## ▶ Un proceso termina:

- ▶ De forma voluntaria (Ej.: a través de llamada exit):
  - ▶ Finalización normal
  - ▶ Finalización con error
- ▶ De forma involuntaria:
  - ▶ Finalizado por el sistema (Ej.: excepción, sin recursos necesarios)
  - ▶ Finalizado por otro proceso (Ej.: a través de llamada al sistema kill)
  - ▶ Finalizado por el usuario (Ej.: control-c por teclado)

- ▶ En Unix/Linux se usan señales como mecanismo
- ▶ Se pueden capturar y tratar (salvo SIGKILL) para evitar finalizar involuntariamente

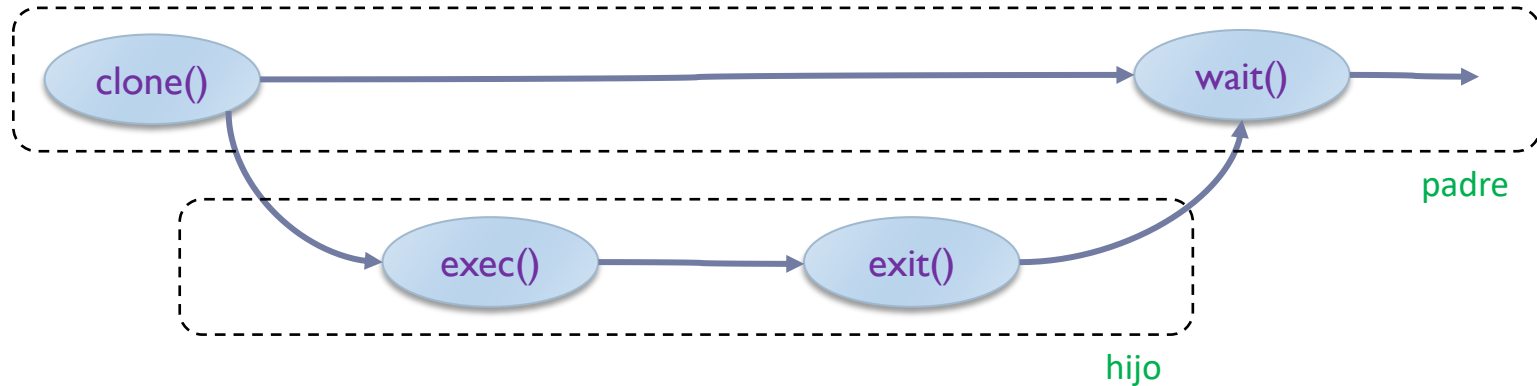
## ▶ Recursos:

- ▶ Todos los recursos se liberan (ficheros, memoria, etc.), >UNIX> salvo el BCP
- ▶ Se notifica al padre, >UNIX> y si hace un wait() entonces se libera el BCP
- ▶ >UNIX> Si padre muere sin hacer un wait pasa a zombie hasta init lo adopta

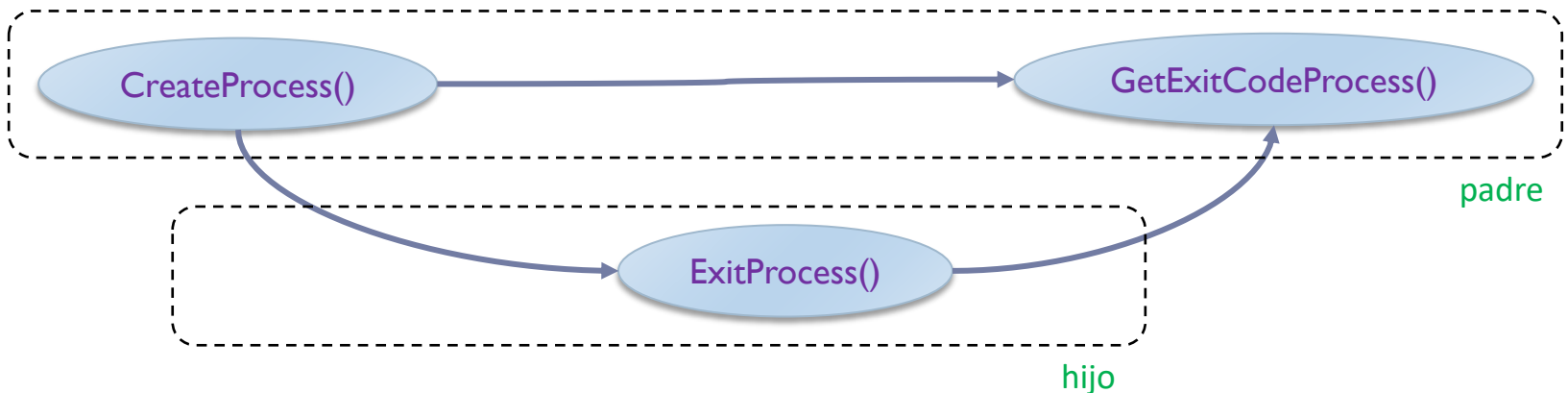
# Creación y terminación de procesos

## Llamadas al sistema

### ► Linux



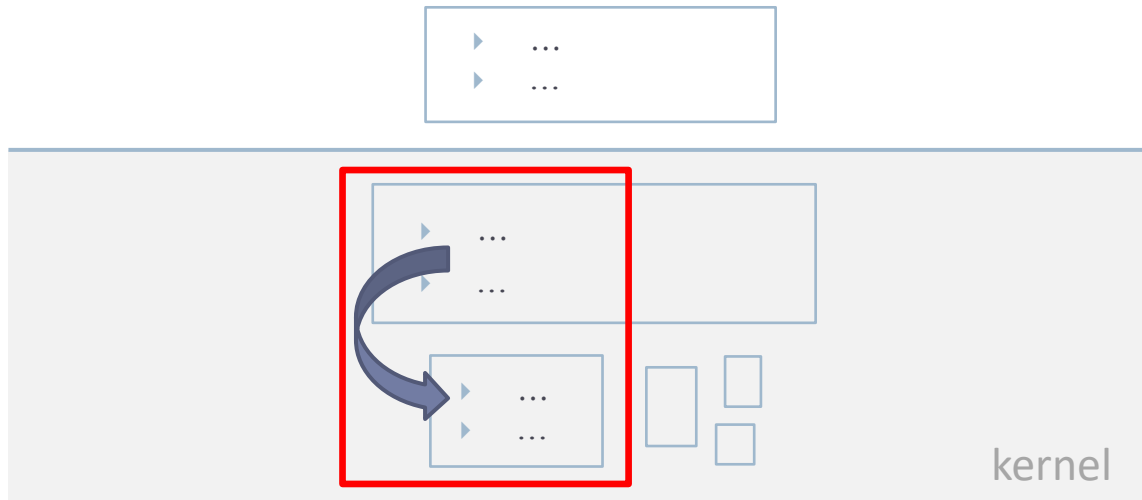
### ► Windows



# Servicios del sistema operativo

## inicialización y finalización de procesos

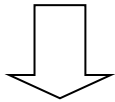
---



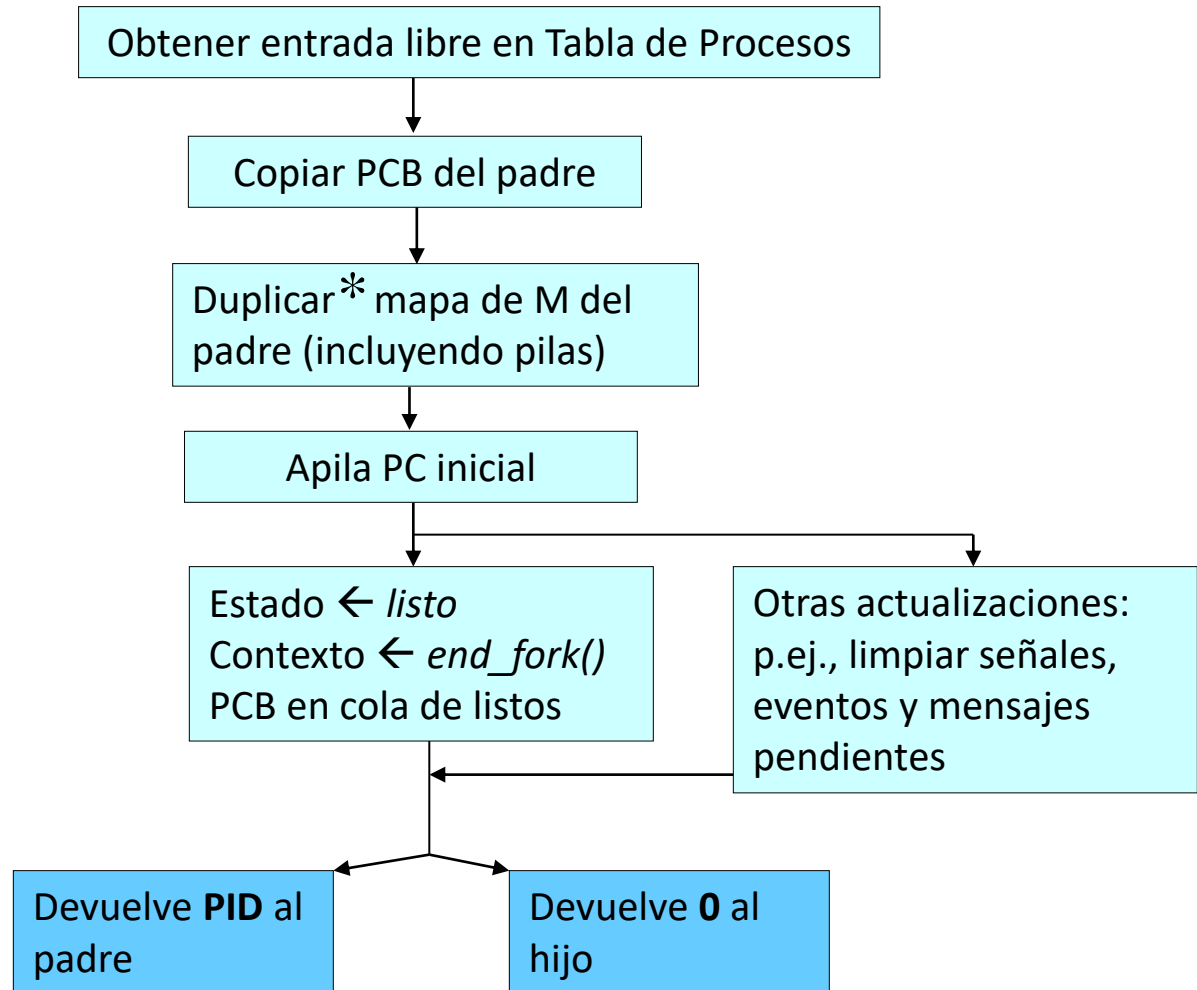
# Creación de procesos

## Linux: clone

clone:



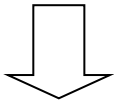
*“Clona al proceso padre y da una nueva identidad al hijo”*



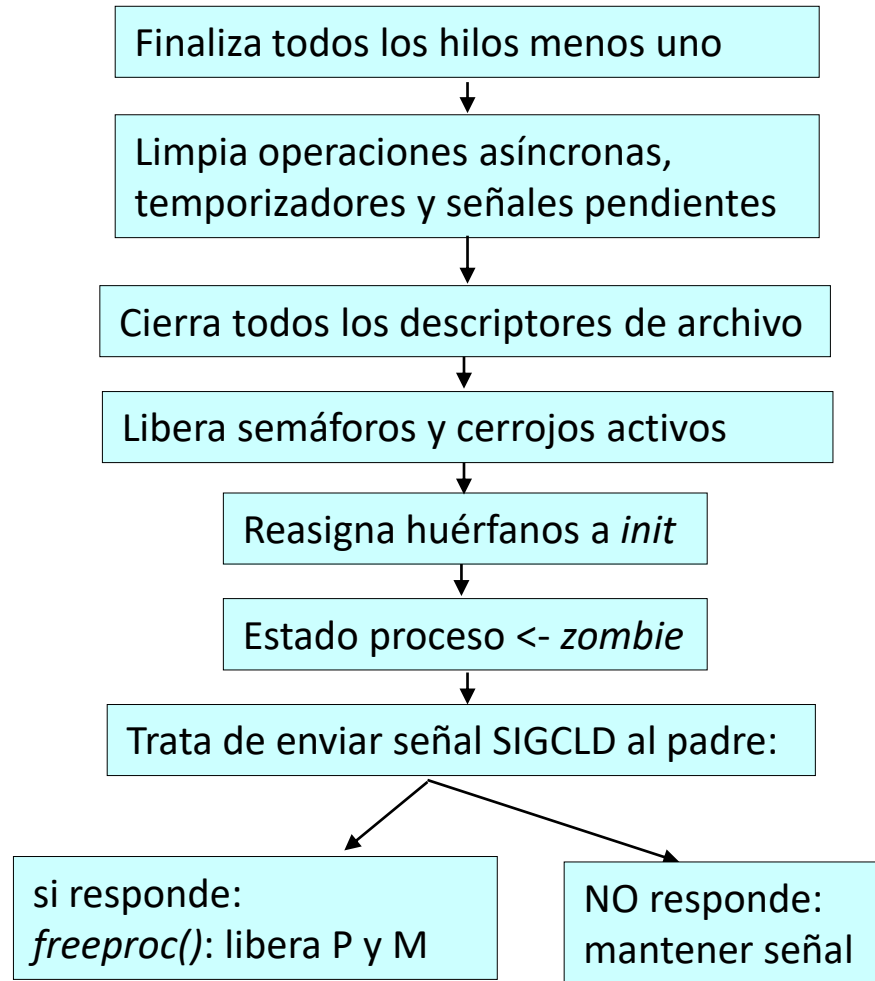
# Terminación de procesos

## Linux: exit

exit:



“Termina la ejecución de un proceso y libera los recursos”

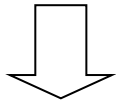




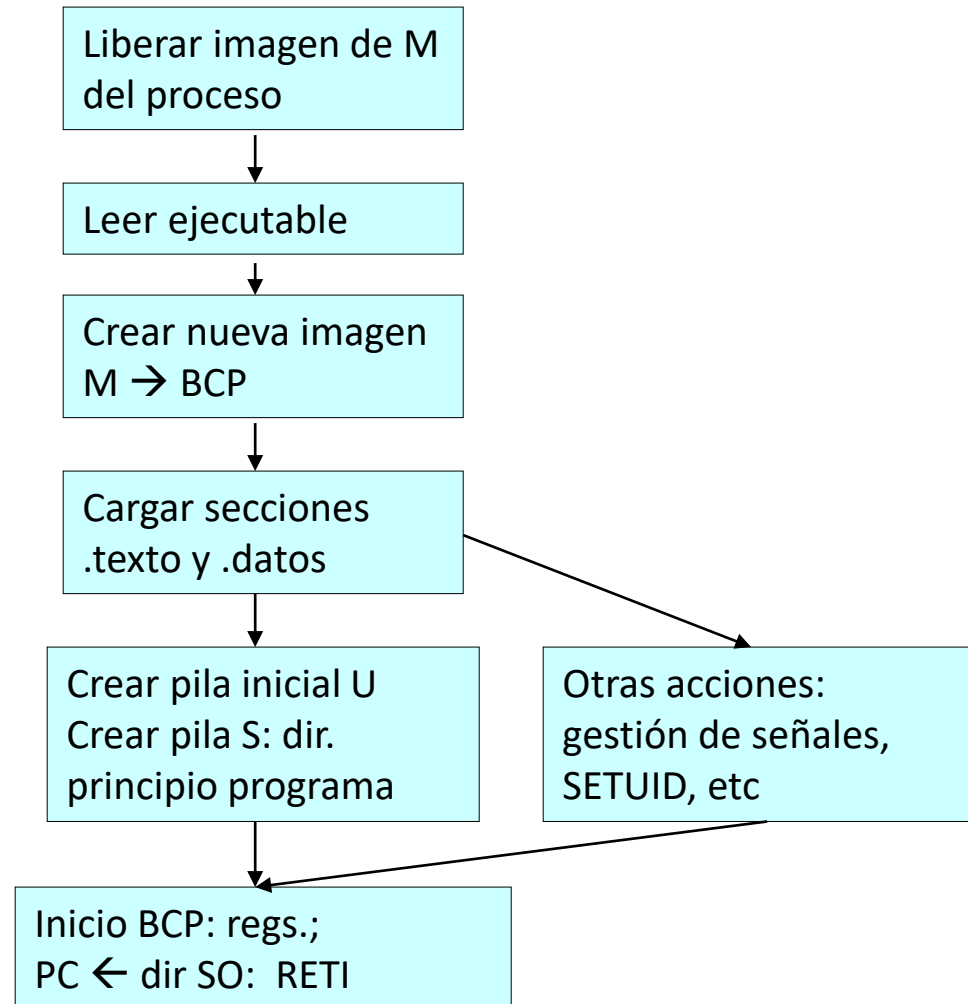
# Cambio de imagen de un proceso

## Linux: exec

exec:



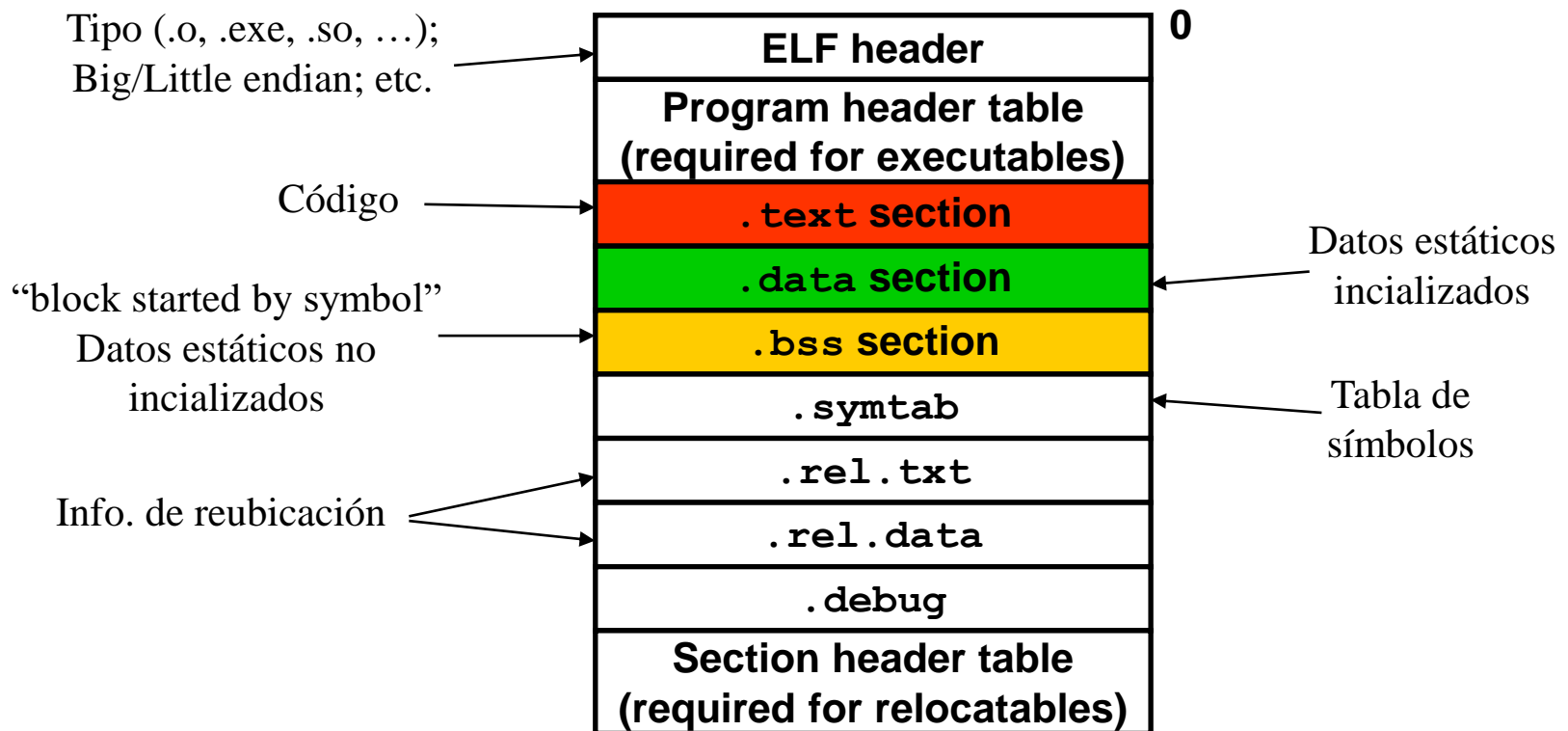
*“Cambia la imagen de memoria de un proceso usando como ‘recipiente’ uno previo”*



# Cambio de imagen de un proceso

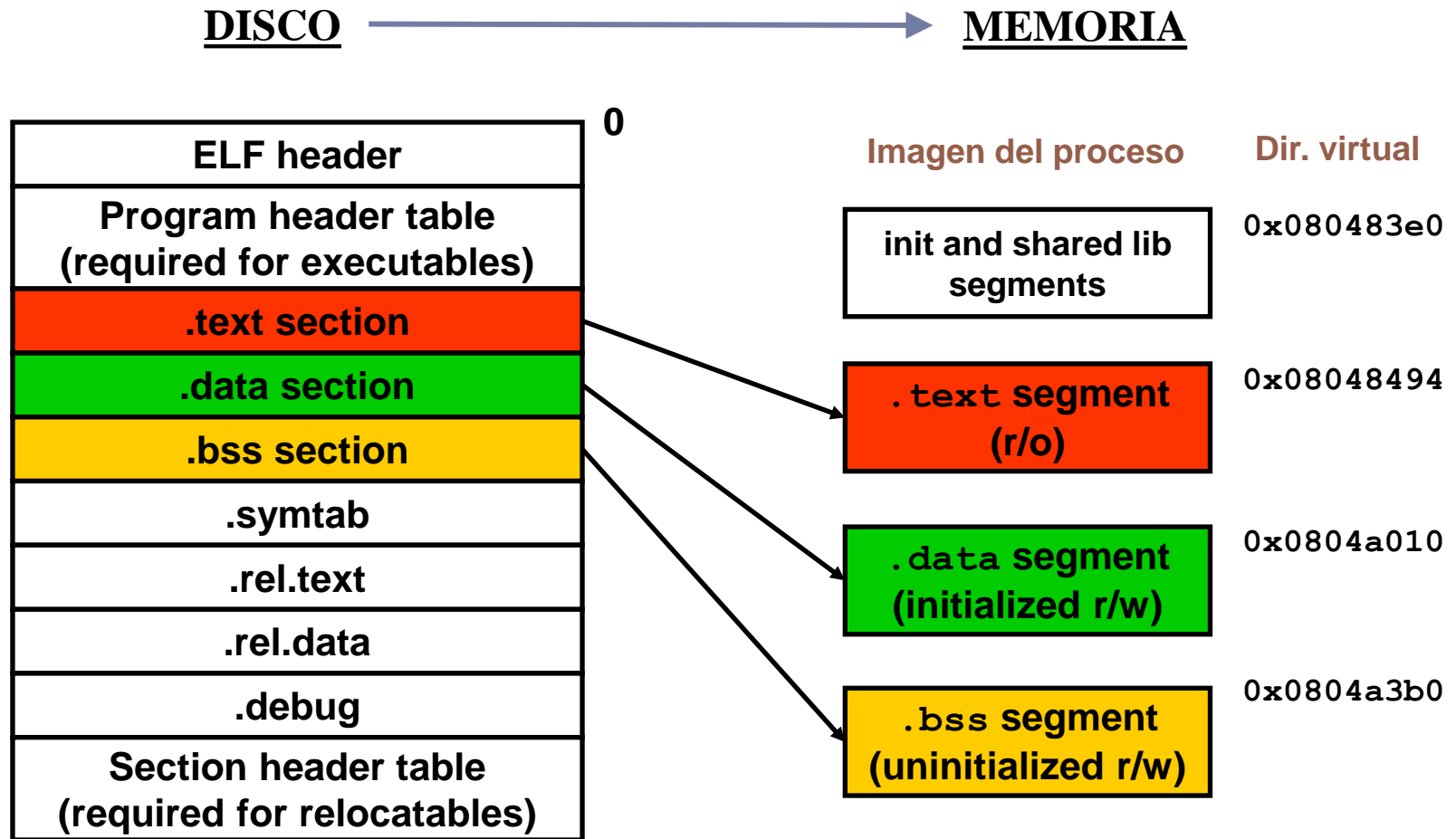
## formato ELF de ejecutable en UNIX

### ► ELF: Executable and Linkable Format



# Cambio de imagen de un proceso

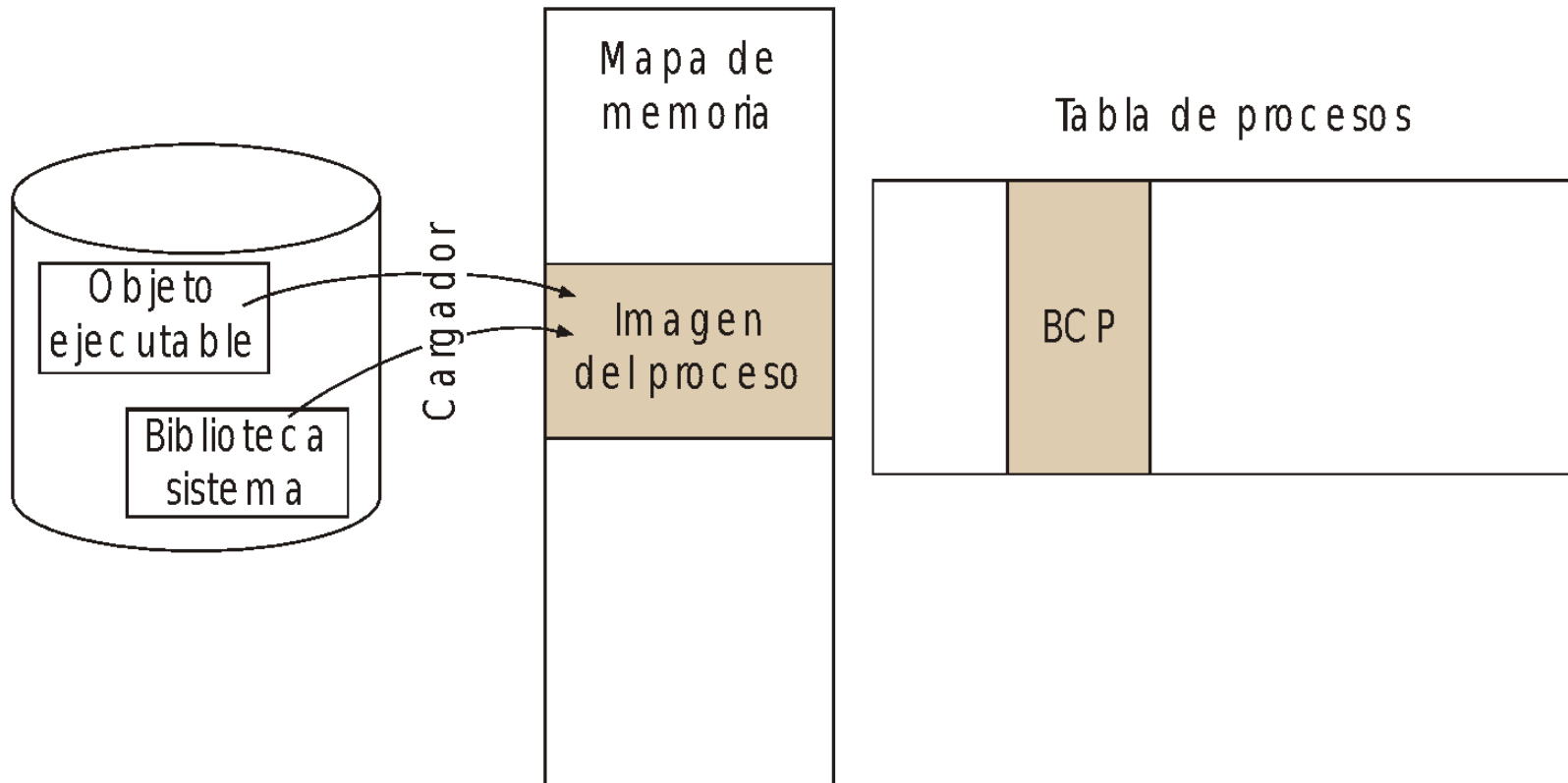
## carga de ejecutable en memoria



# Cambio de imagen de un proceso

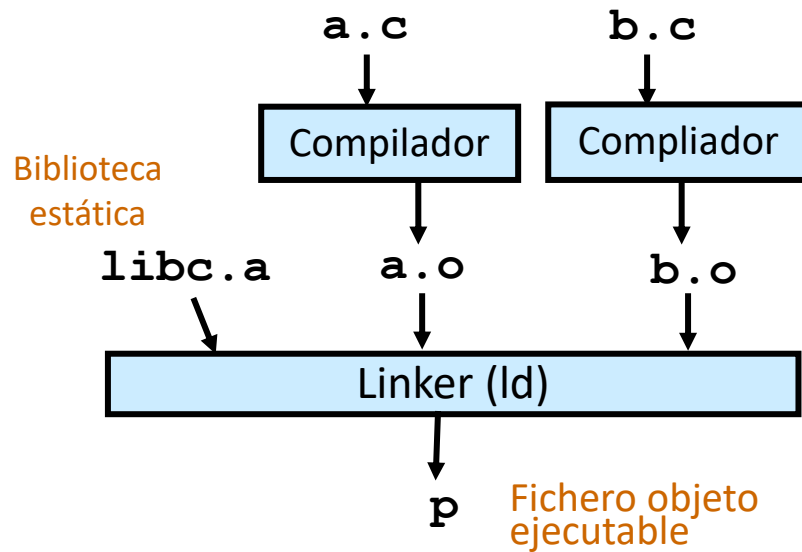
## carga de ejecutable en memoria

---

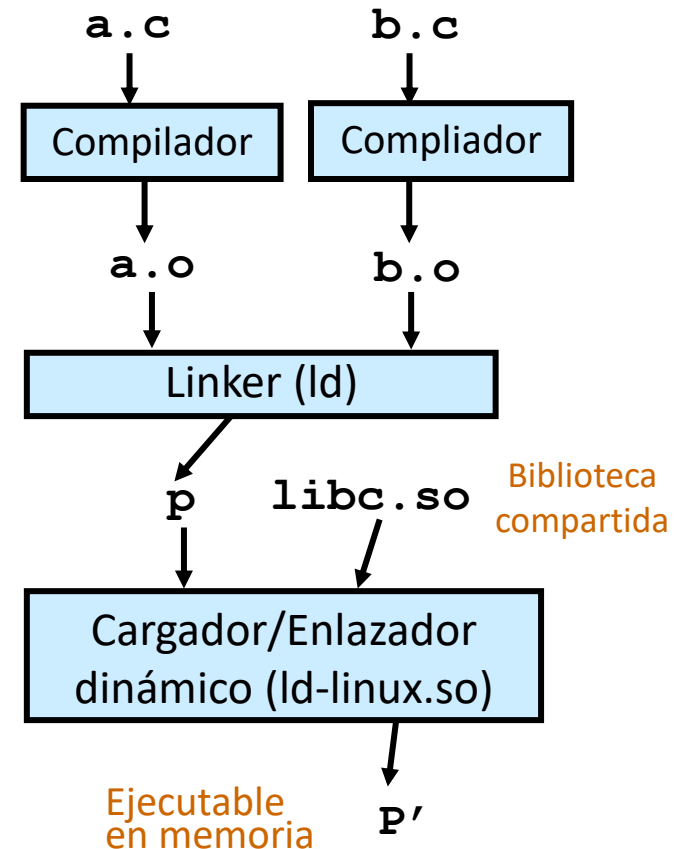


# Cambio de imagen de un proceso

## generación de ejecutable



Bibliotecas estáticas



Bibliotecas dinámicas

# Lección 3

## Procesos e hilos

Sistemas Operativos  
Ingeniería Informática