Grupo ARCOS
Universidad Carlos III de Madrid

Lección 3
Procesos e hilos

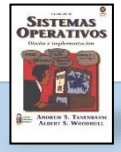
Sistemas Operativos
Ingeniería Informática

Lecturas recomendadas

Base

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

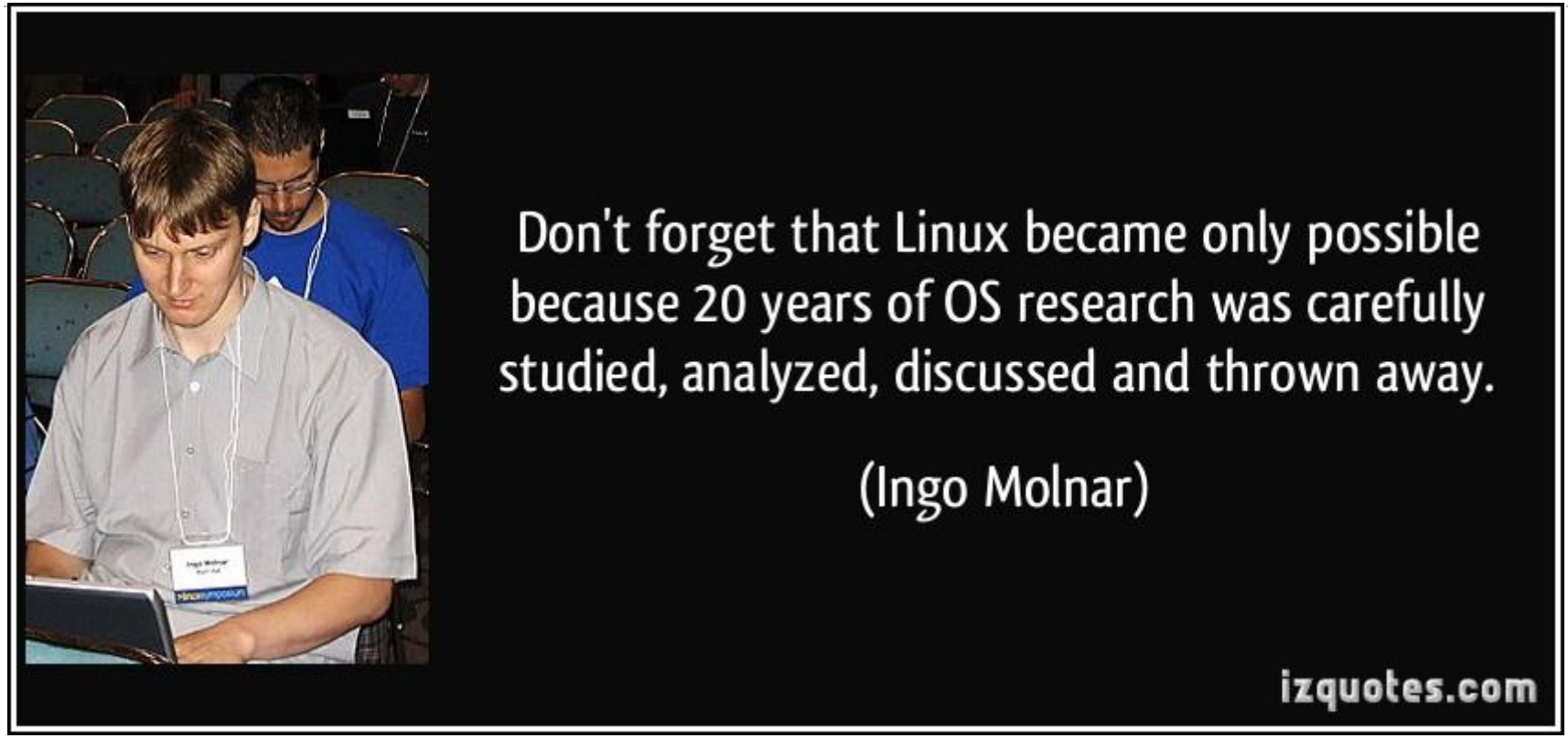
Recomendada



1. Tanenbaum 2006(en):
   1. Cap.3
2. Stallings 2005:
   1. Parte tres
3. Silberschatz 2006:
   1. Cap. 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

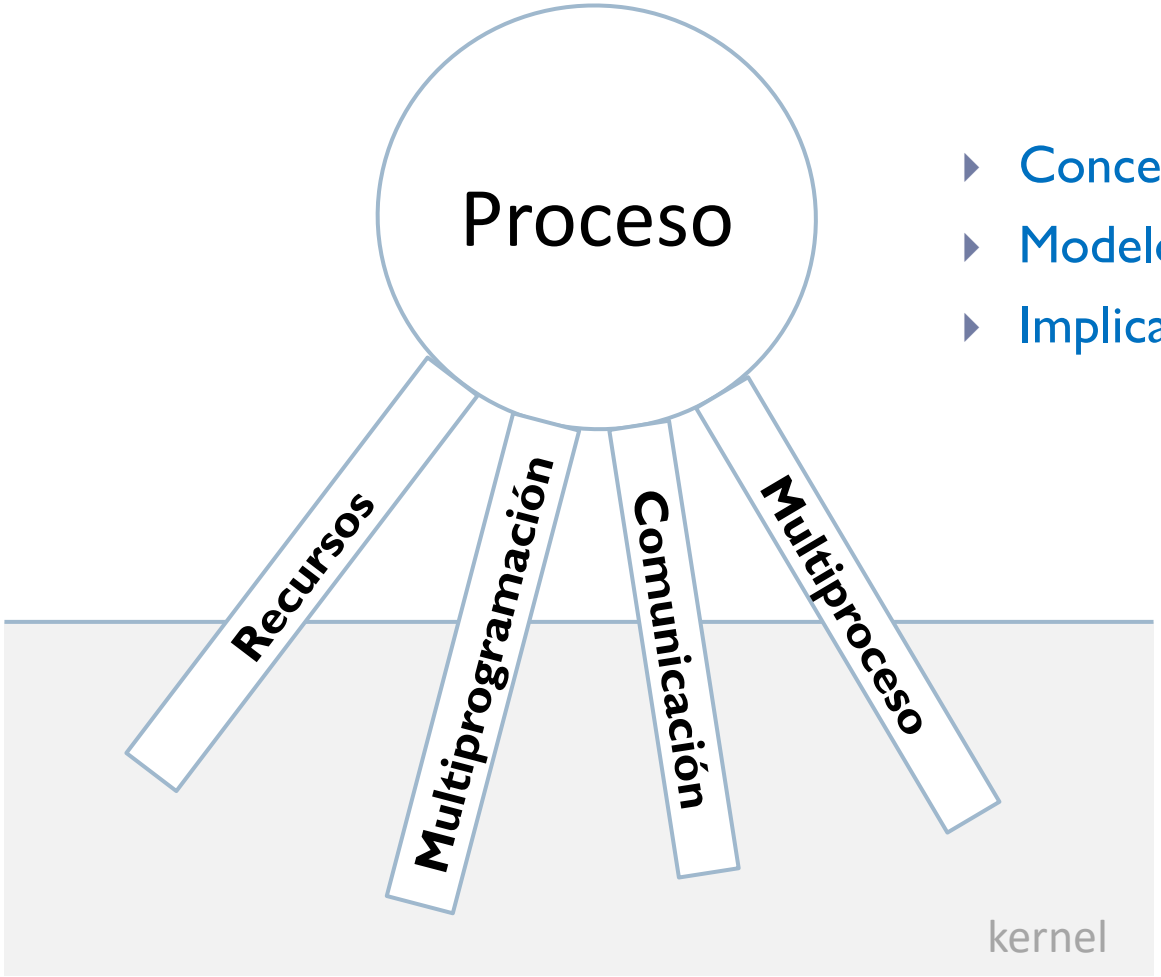
Contenidos

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.
   - Estructura de datos de procesos e hilos en el núcleo
   - Diseño e implementación de la multiprogramación y la multitarea en el núcleo

Contenidos

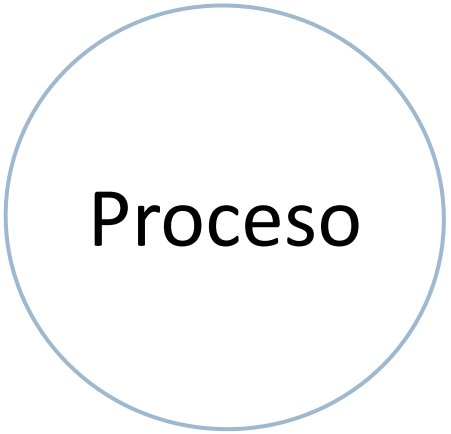
1. **Introducción**
   - **Definición de proceso.**
   - **Modelo ofrecido: recursos, multiprogramación, multitarea y multiproceso**
2. Ciclo de vida del proceso: estado de procesos.
3. Servicios para gestionar procesos que da el sistema operativo.
4. Definición de hilo o *thread*
5. Hilos de biblioteca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

Introducción



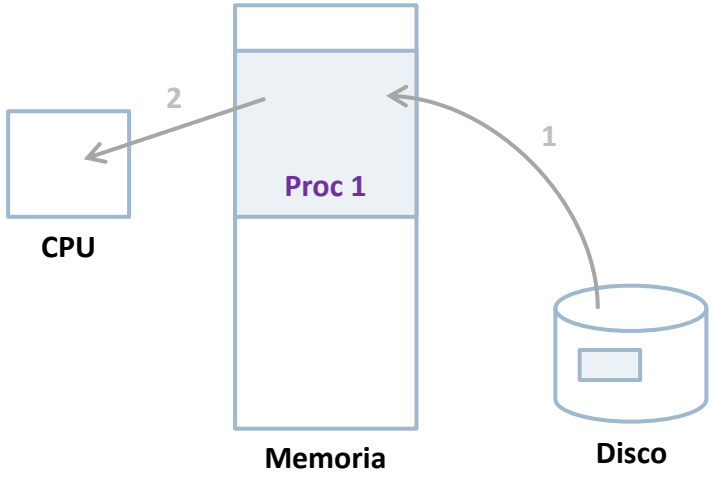
- Concepto de proceso
- Modelo ofrecido
- Implicaciones en S.O.

Introducción



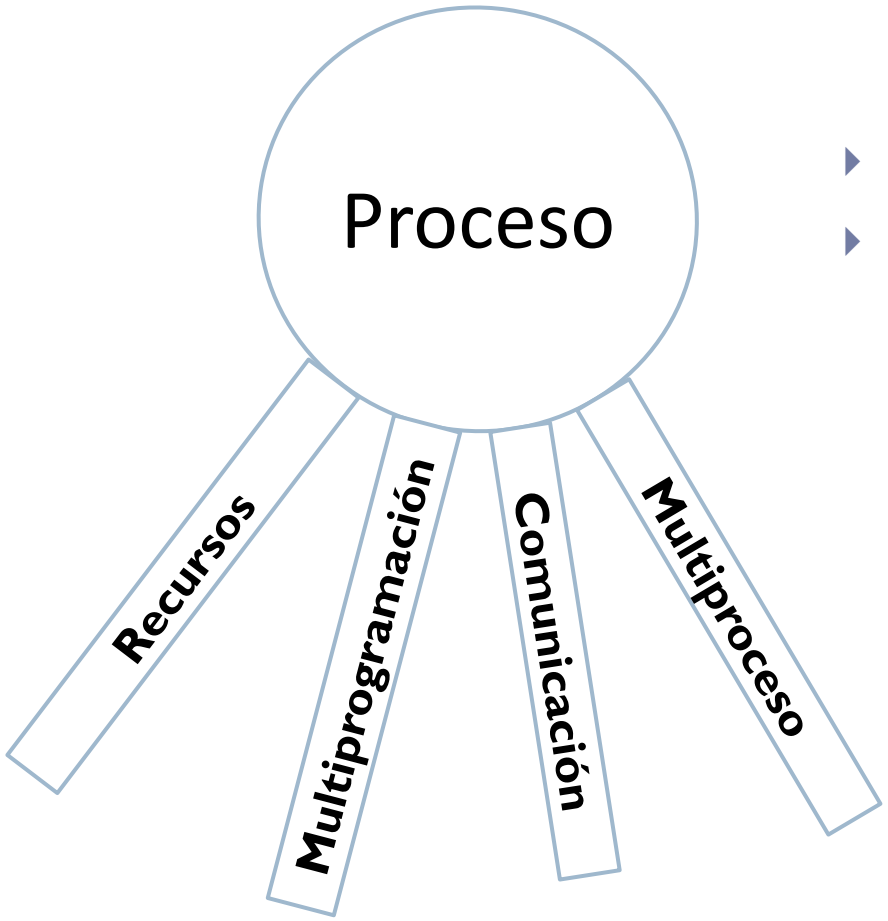
- **Concepto de proceso**

Concepto de proceso



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

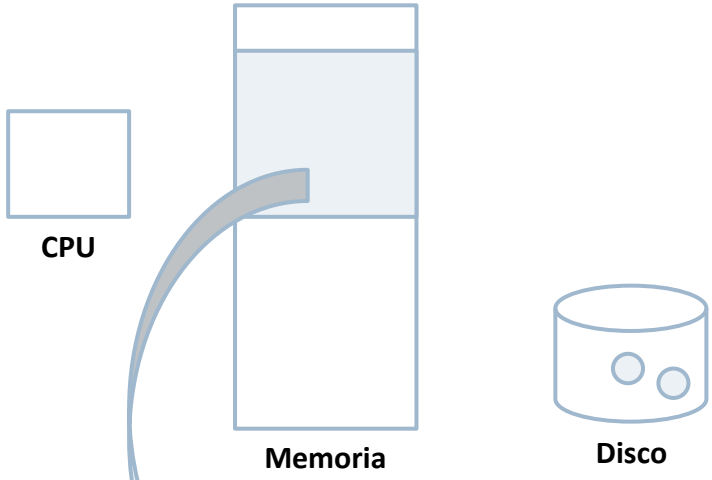
Introducción



- Concepto de proceso
- **Modelo ofrecido**

Modelo ofrecido

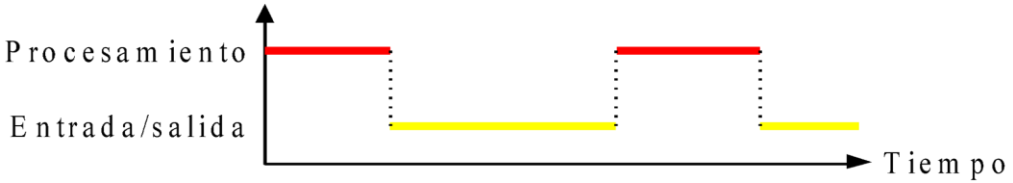
- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- multitarea
- multiproceso



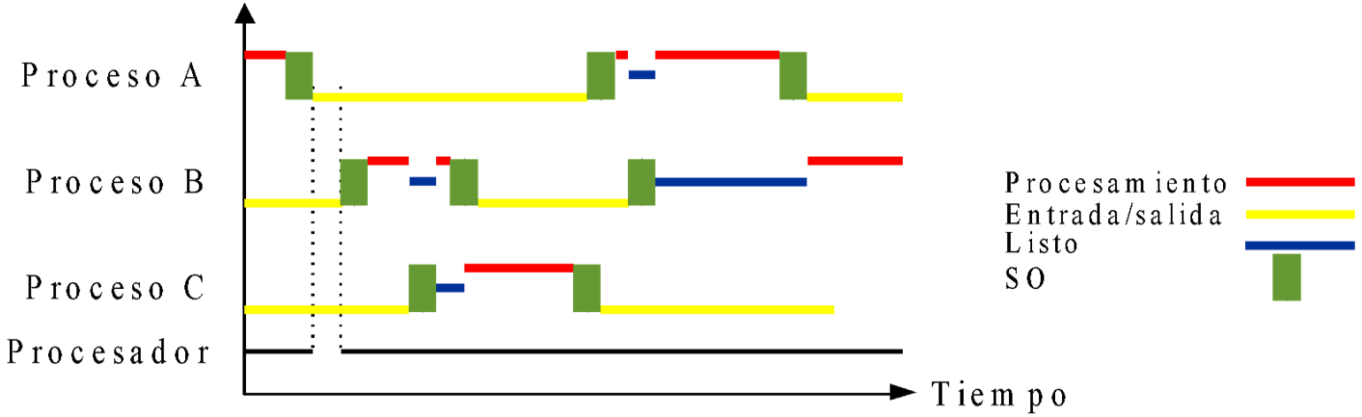
- Recursos asociados
  - Zonas de memoria
    - Al menos: código, datos y pila
  - Archivos abiertos
  - Señales

Principios de la multitarea… (multiprogramación)

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

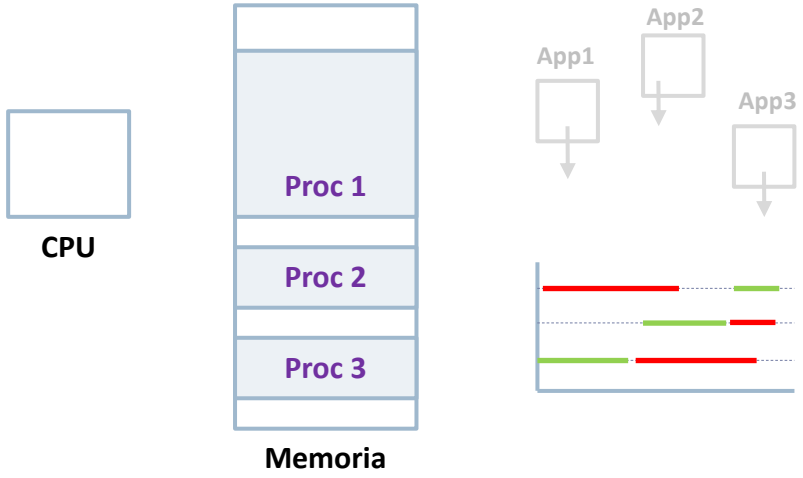


- La memoria almacena varios procesos.
- Paralelismo real entre E/S y CPU/UCP (DMA).



Modelo ofrecido

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

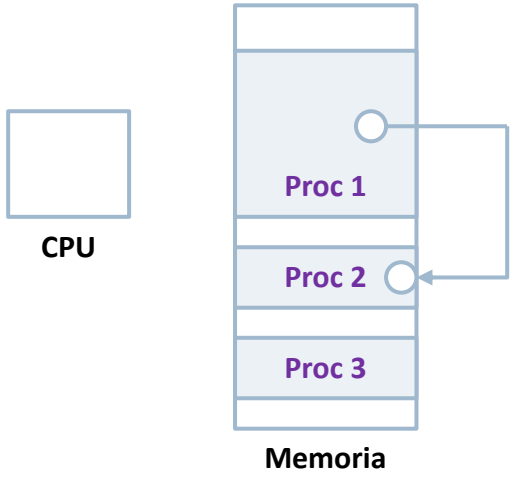


Multiprogramación

- 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

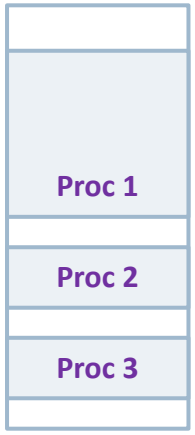
- recursos
- multiprogramación
  - **protección/compartición**
  - jerarquía de procesos
- multitarea
- multiproceso

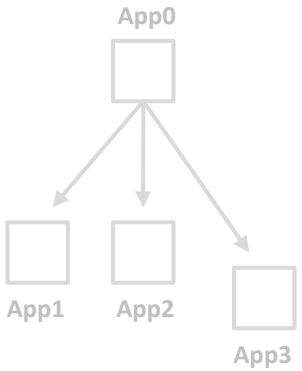


- Protección / Compartición
  - El espacio de direcciones privado por aplicación, pero
  - Posibilidad de comunicar datos entre dos aplicaciones
    - Paso de mensajes
    - Compartición de memoria

Modelo ofrecido

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

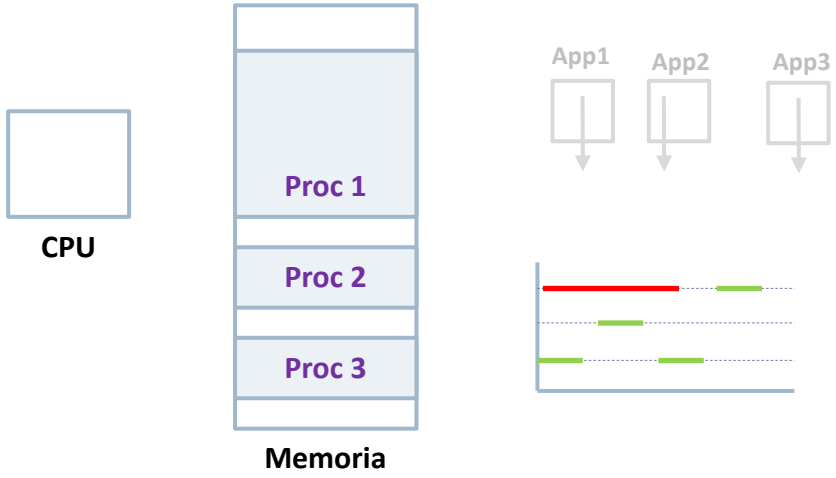




- Jerarquía de procesos
  - Creación de proceso
    - Como copia de otro proceso existente
    - A partir del programa en disco
    - Como proceso en el arranque
  - Grupo de procesos que comparten mismo tratamiento

Modelo ofrecido

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

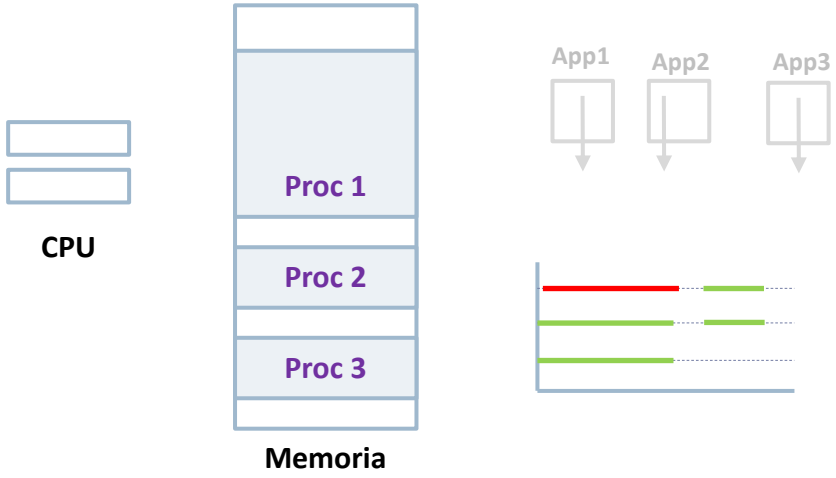


Multitarea

- 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**

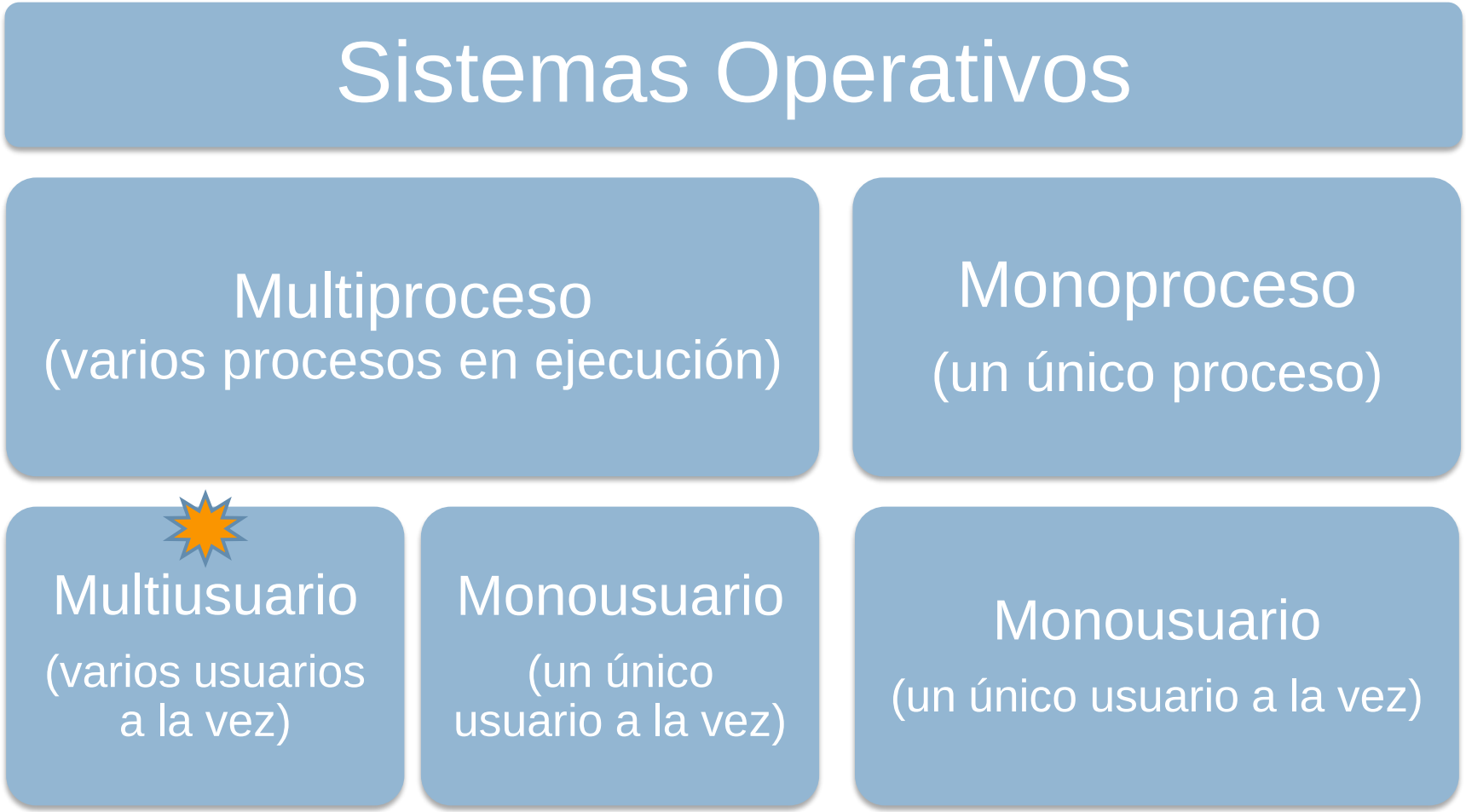


Multiproceso

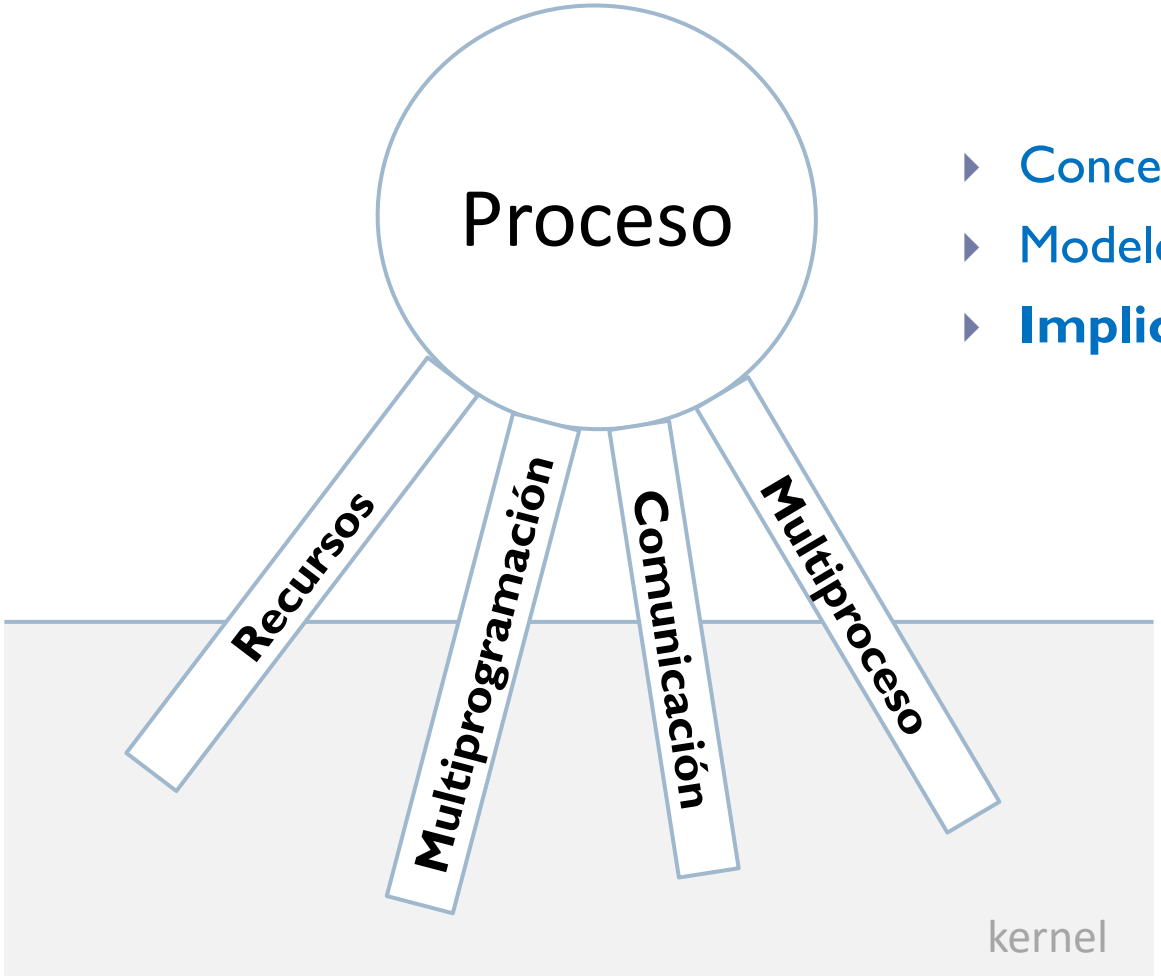
- Se dispone de varios procesadores (multicore/multiprocesador)
- Además del reparto de cada CPU (multitarea) hay paralelismo real entre varias tareas (tantas como procesadores)
  - Se suele usar planificador y estructuras de datos separadas por procesador con algún mecanismo de equilibrio de carga

Tipos de sistemas operativos

modelo a medida



Introducción



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

Implicaciones en el sistema operativo

1. Estructuras de datos

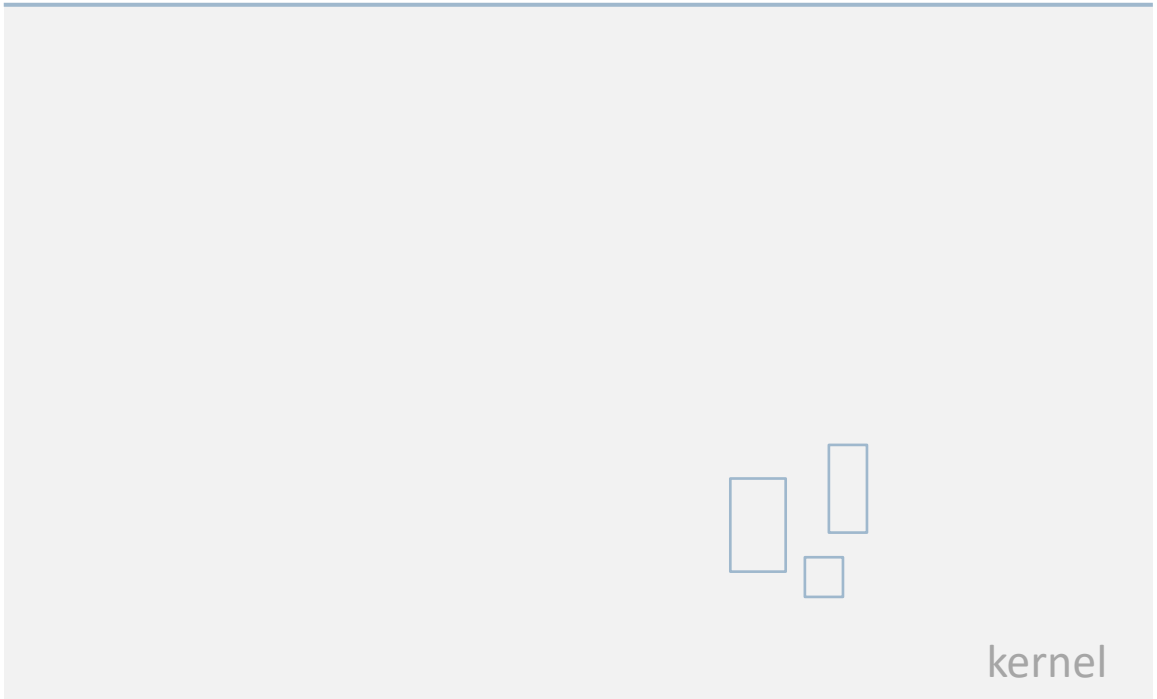
Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	• Zonas de memoria (código, datos y pila) • Archivos abiertos • Señales activas	• Diversas funciones internas • Diversas funciones de servicio para memoria, ficheros, etc.
Multiprogramación	• Estado de ejecución • Contexto: registros de CPU… • Lista de procesos	• Int. hw/sw de dispositivos • Planificador • Crear/Destruir/Planificar proceso

○ Protección / Compartición	• Paso de mensajes
	  • Cola de mensajes de recepción
	• Memoria compartida
	  • Zonas, locks y conditions
		• Envío/Recepción mensaje y gestión de la cola de mensaje
		• API concurrencia y gestión de estructuras de datos

○ Jerarquía de procesos	• Relación de parentesco • Conjuntos de procesos relacionados • Procesos de una misma sesión	• Clonar/Cambiar imagen de proceso • Asociar procesos e indicar proceso representante
Multitarea	• Quantum restante • Prioridad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso
Multiproceso	• Afinidad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso

Implicaciones en el sistema operativo

1. Estructuras de datos



Implicaciones en el sistema operativo

2. Funciones: de gestión internas

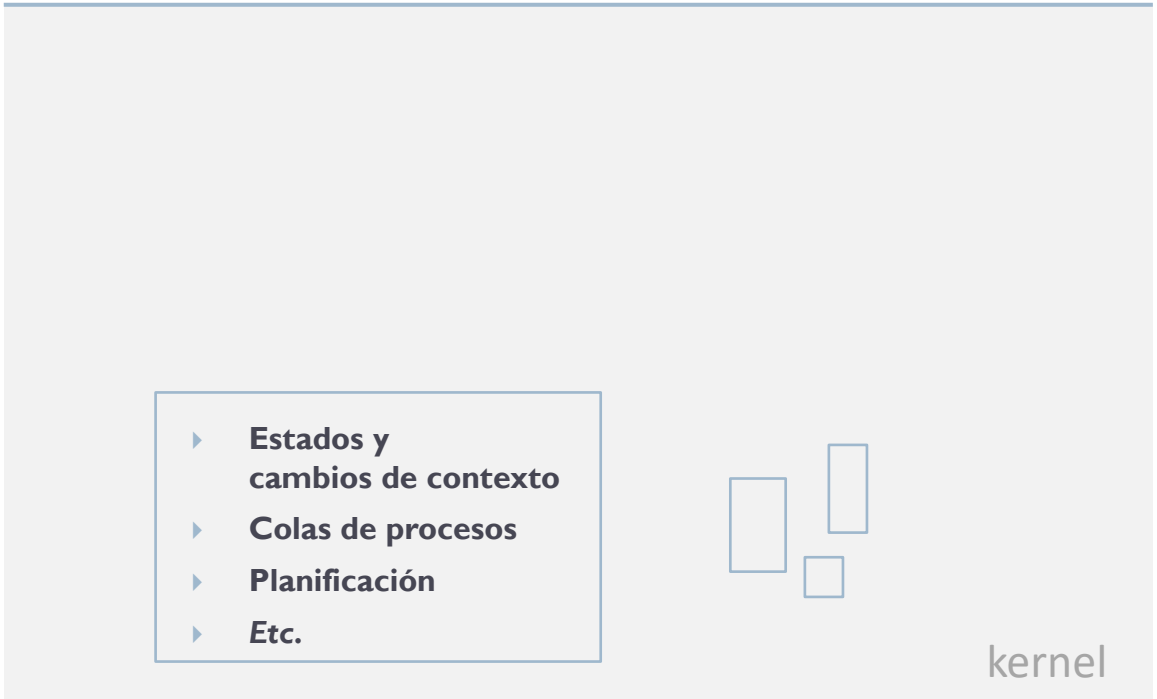
Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	• Zonas de memoria (código, datos y pila) • Archivos abiertos • Señales activas	• Diversas funciones internas • Diversas funciones de servicio para memoria, ficheros, etc.
Multiprogramación	• Estado de ejecución • Contexto: registros de CPU… • Lista de procesos	• Int. hw/sw de dispositivos • Planificador • Crear/Destruir/Planificar proceso

○ Protección / Compartición	• Paso de mensajes
	  • Cola de mensajes de recepción
	• Memoria compartida
	  • Zonas, locks y conditions
		• Envío/Recepción mensaje y gestión de la cola de mensaje
		• API concurrencia y gestión de estructuras de datos

○ Jerarquía de procesos	• Relación de parentesco • Conjuntos de procesos relacionados • Procesos de una misma sesión	• Clonar/Cambiar imagen de proceso • Asociar procesos e indicar proceso representante
Multitarea	• Quantum restante • Prioridad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso
Multiproceso	• Afinidad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso

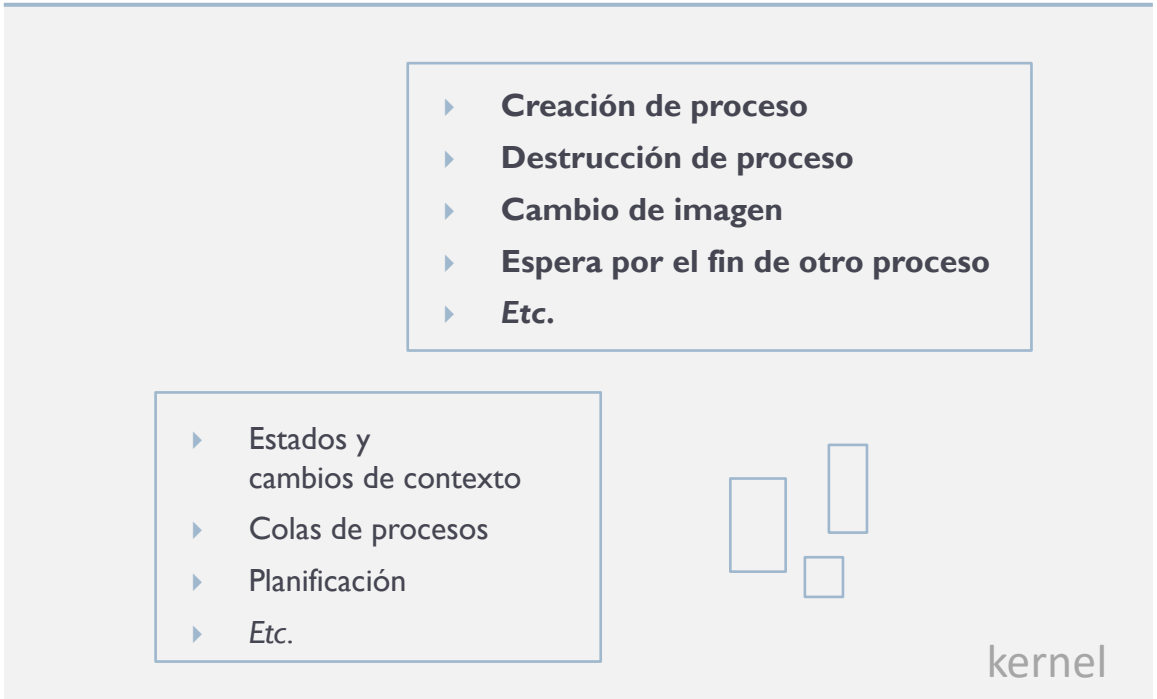
Implicaciones en el sistema operativo

2. Funciones: de gestión internas



Implicaciones en el sistema operativo

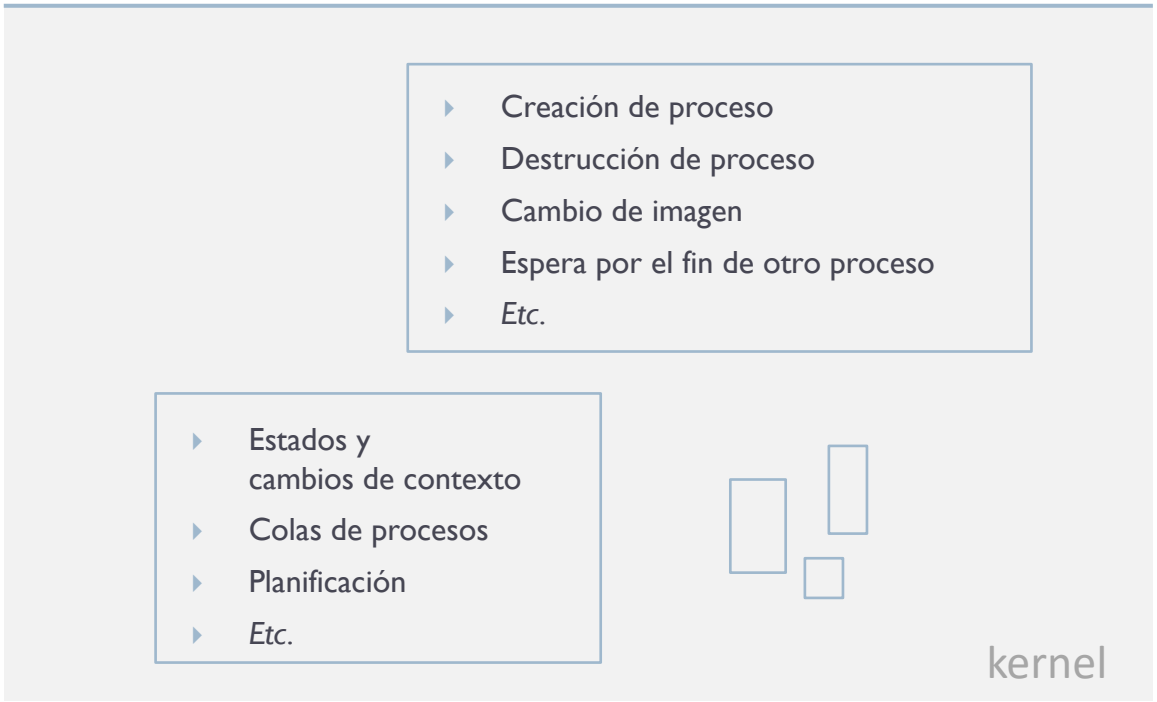
3. Funciones: de servicio



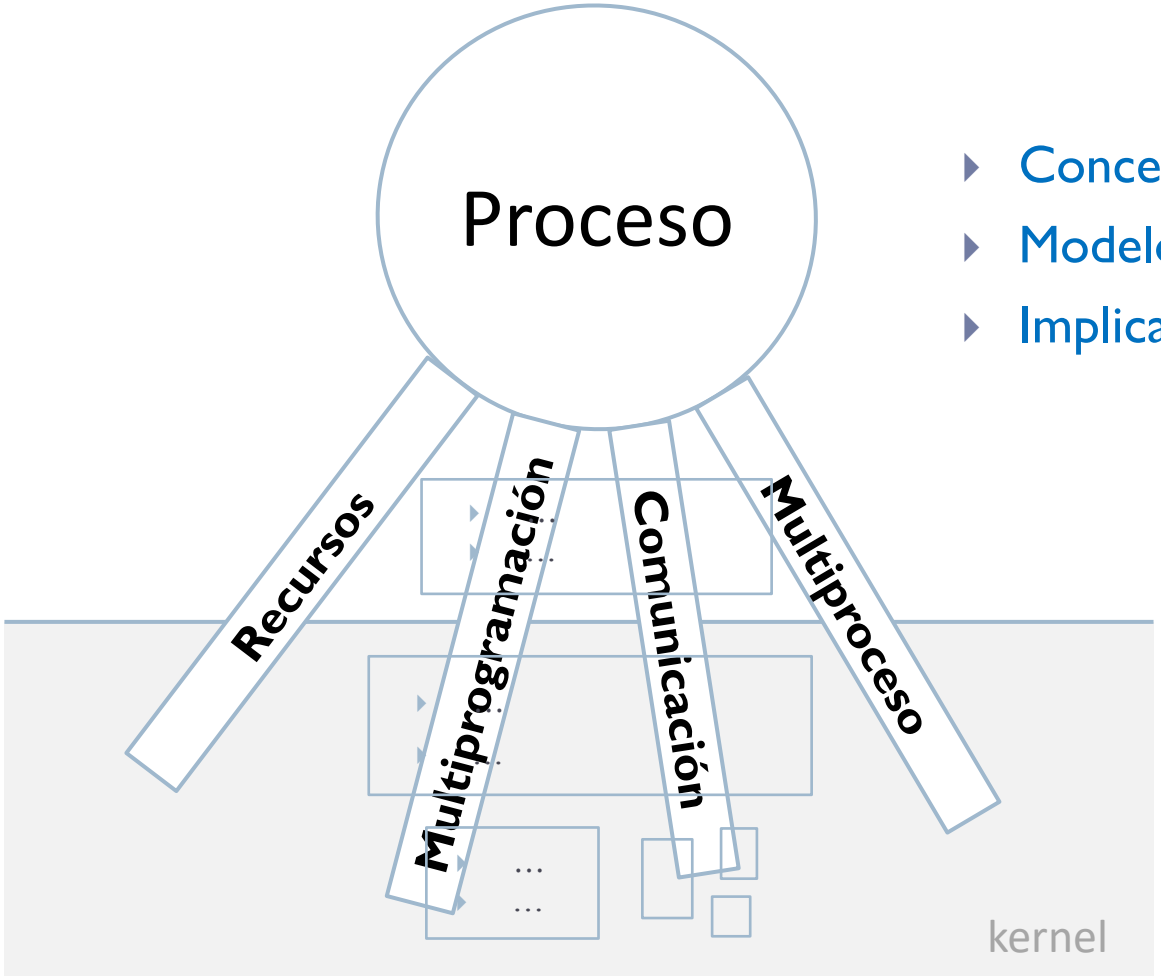
Implicaciones en el sistema operativo

3. Funciones: API de servicio

- fork, exit, exec, wait, …
- *pthread_create, pthread…*

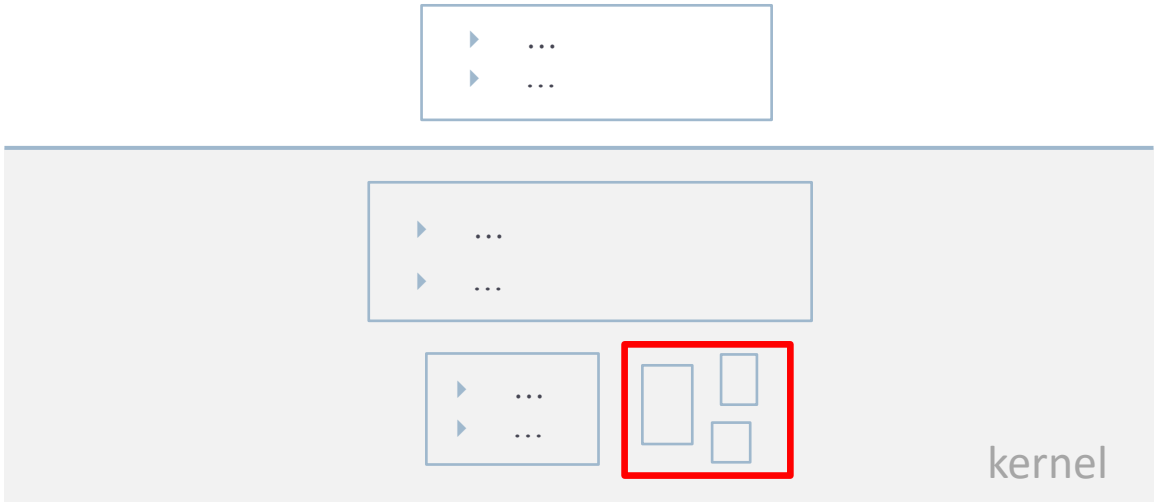


Introducción
resumen

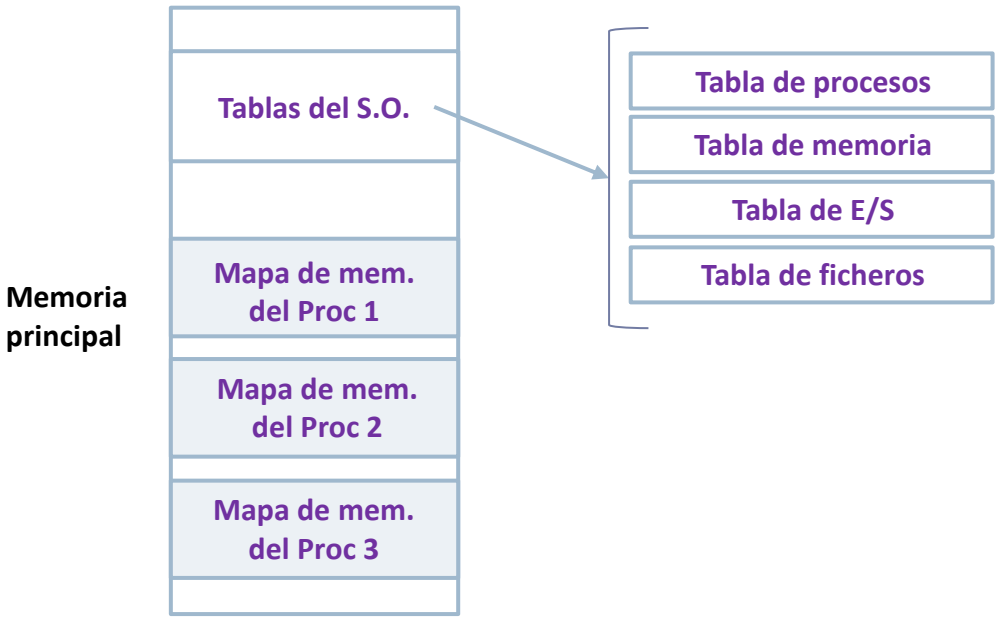


- Concepto de proceso
- Modelo ofrecido
- Implicaciones en S.O.

Principales estructuras de datos

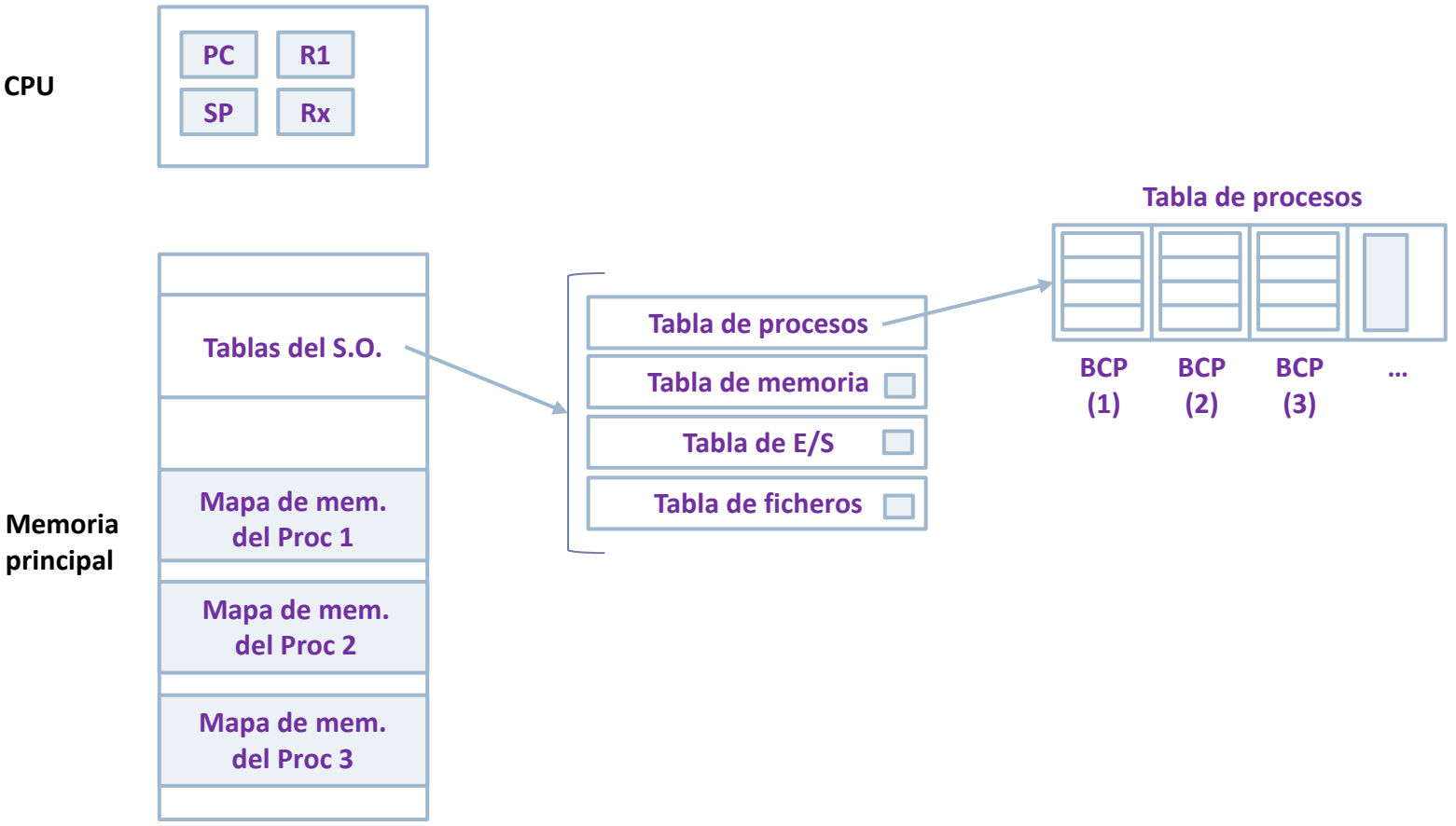


Información en el sistema operativo



Información para un proceso está:

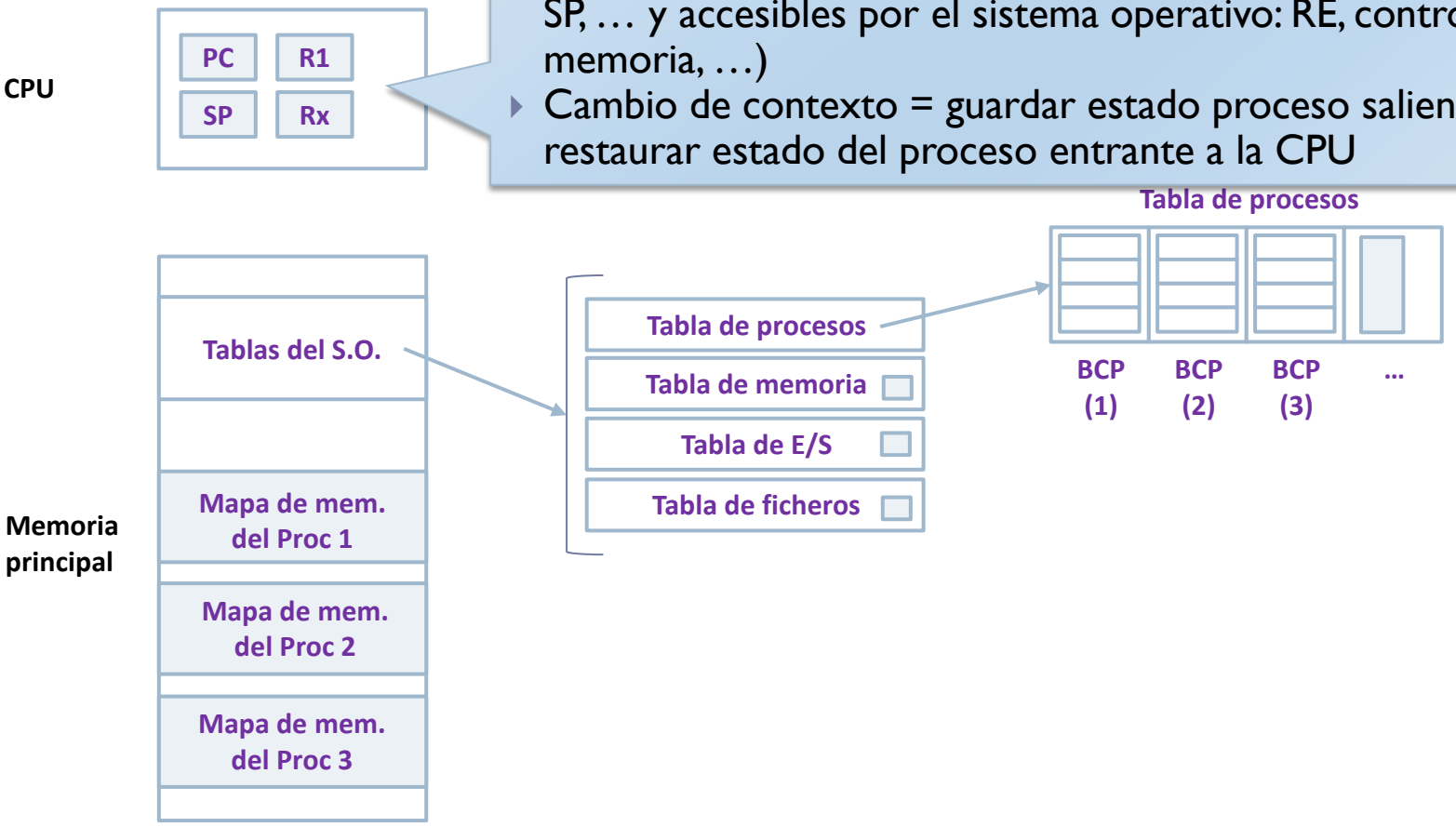
en procesador + en memoria + datos adicionales del S.O.



Información para un proceso

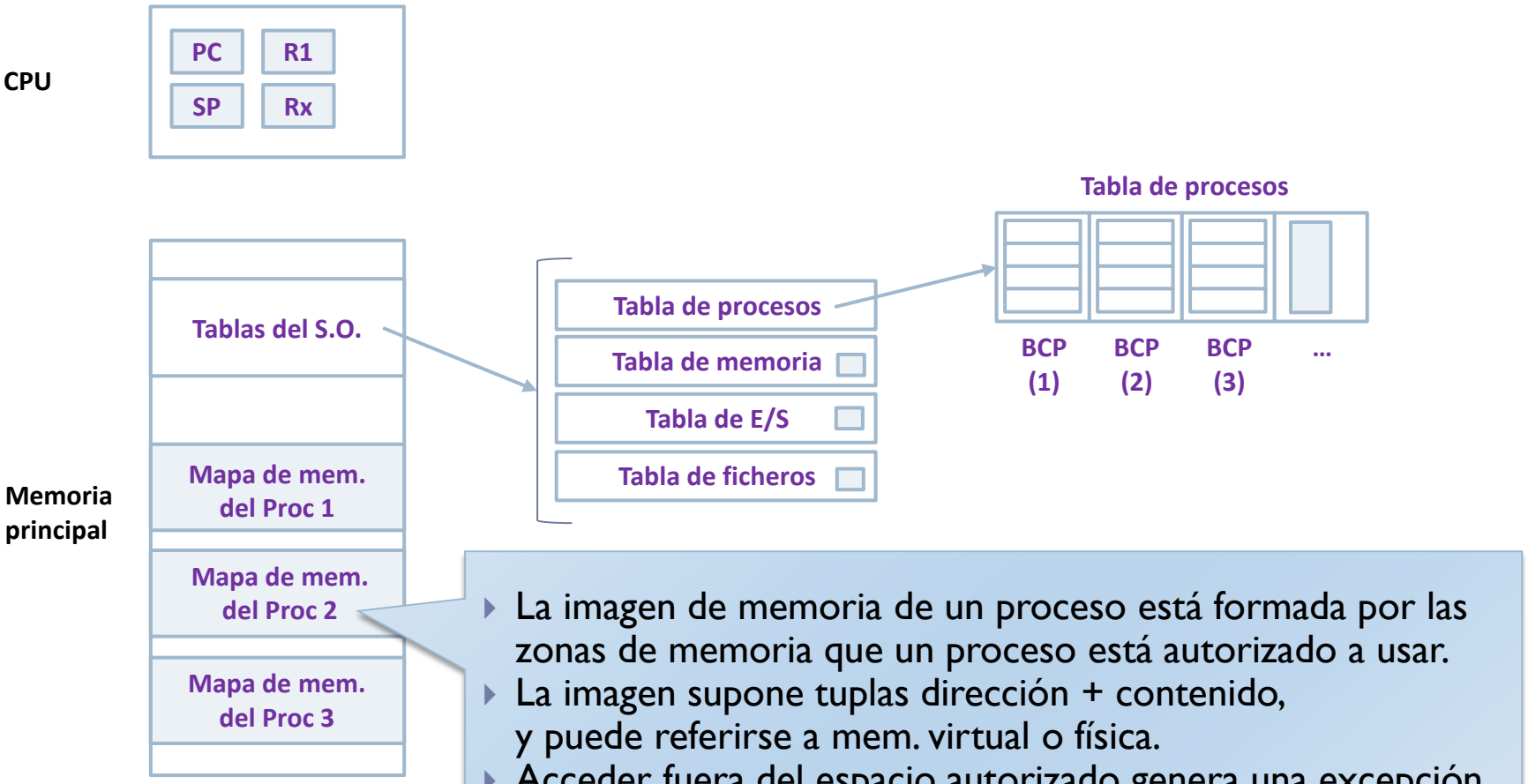
estado del procesador

- **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



Información para un proceso

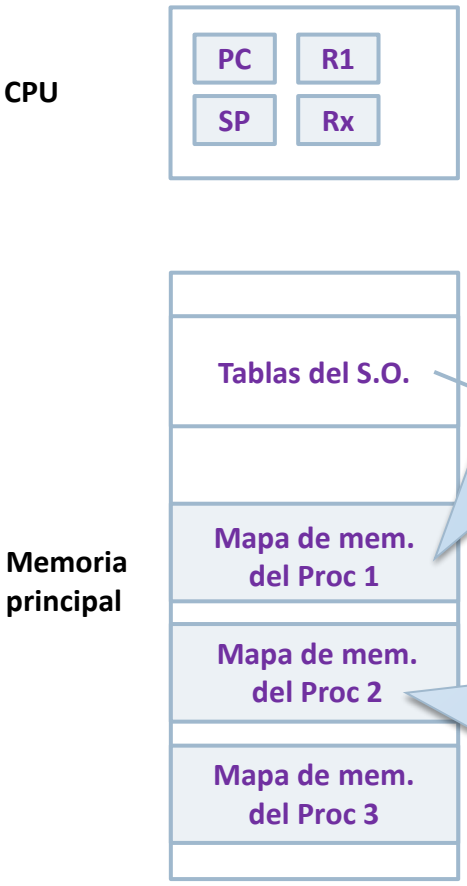
imagen de memoria



- 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

imagen de memoria



	… de tamaño fijo	… tamaño variable

Proceso con única región…	Usado en sistemas sin memoria virtual	▸ Sistemas sin memoria virtual:
		  ▸ Necesita espacio de reserva ➔ Desperdicio de memoria.
		▸ Sistemas con memoria virtual:
		  ▸ Espacio de reserva virtual ➔ Factible, menos flexible que múltiples regiones.

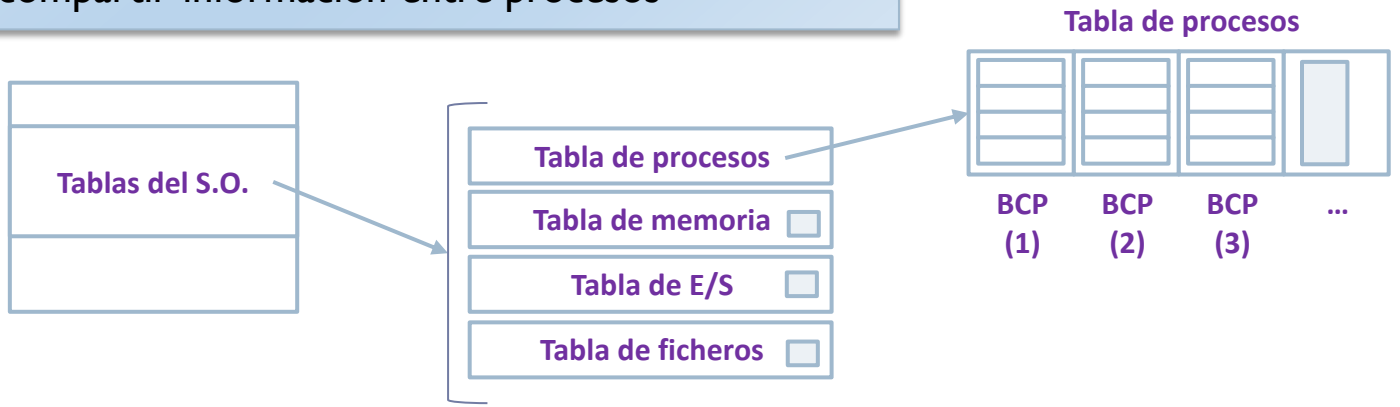
Proceso con número fijo de regiones…		▸ Regiones prefijadas (texto, datos, pila). ▸ Cada región puede crecer. ▸ Con memoria virtual el hueco entre pila y datos no consume recursos físicos

Proceso con número variable de regiones…		▸ Un proceso se estructura en un número arbitrario de regiones (más actual).
		▸ Más avanzado y muy flexible:
		  ▸ Regiones compartidas.
		  ▸ Regiones con distintos permisos.

- 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

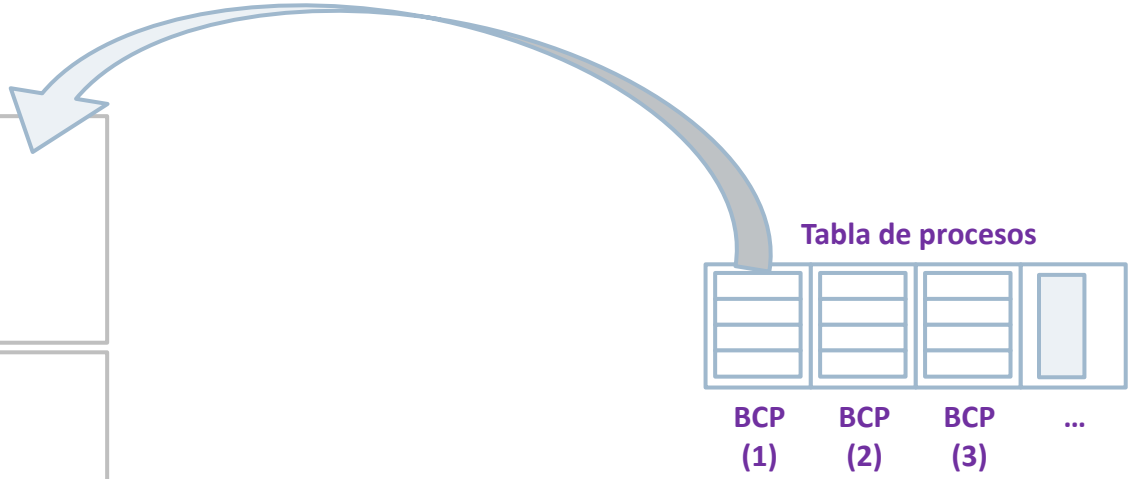
BCP: entrada de la tabla de procesos

- Gestión de proceso

estado
- Registros generales
- Contador de programa
- Registro de estado
- Puntero de pila

Id.
- Identificador del proceso
- Proceso padre
- Grupo de proceso

control
- Prioridad
- Parámetros del planificador
- Señales
- Instante inicio de ejecución
- Tiempo de uso de CPU
- Tiempo hasta siguiente alarma



- *Process Control Block* (PCB / BCP)
  - Estructura de datos con la información necesaria para gestionar un proceso en particular
  - Manifestación de un proceso en el kernel
- *Thread Control Block* (TCB / BCT)
  - Similar al BCP para cada hilo de un proceso

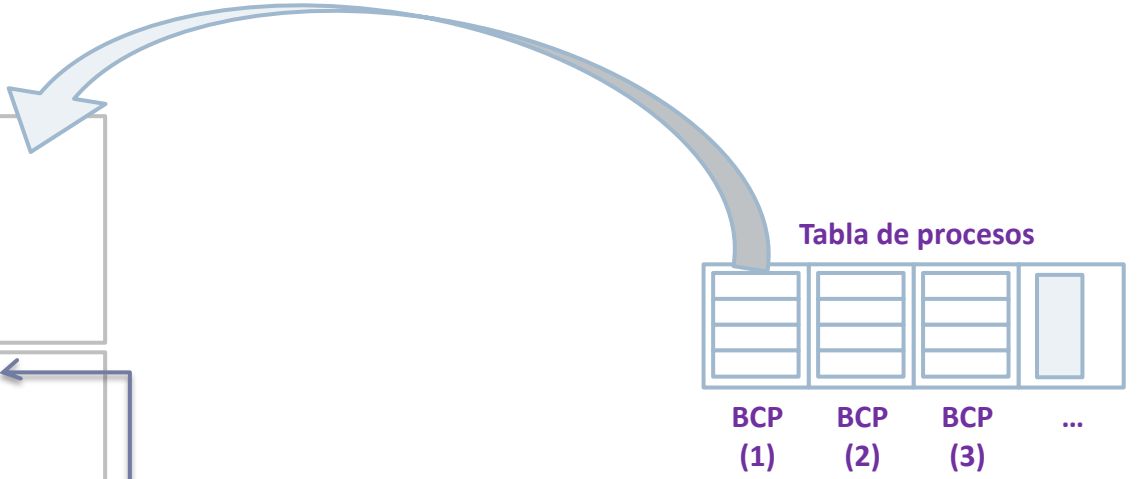
BCP: entrada de la tabla de procesos

- Gestión de proceso

estado
- Registros generales
- Contador de programa
- Registro de estado
- Puntero de pila

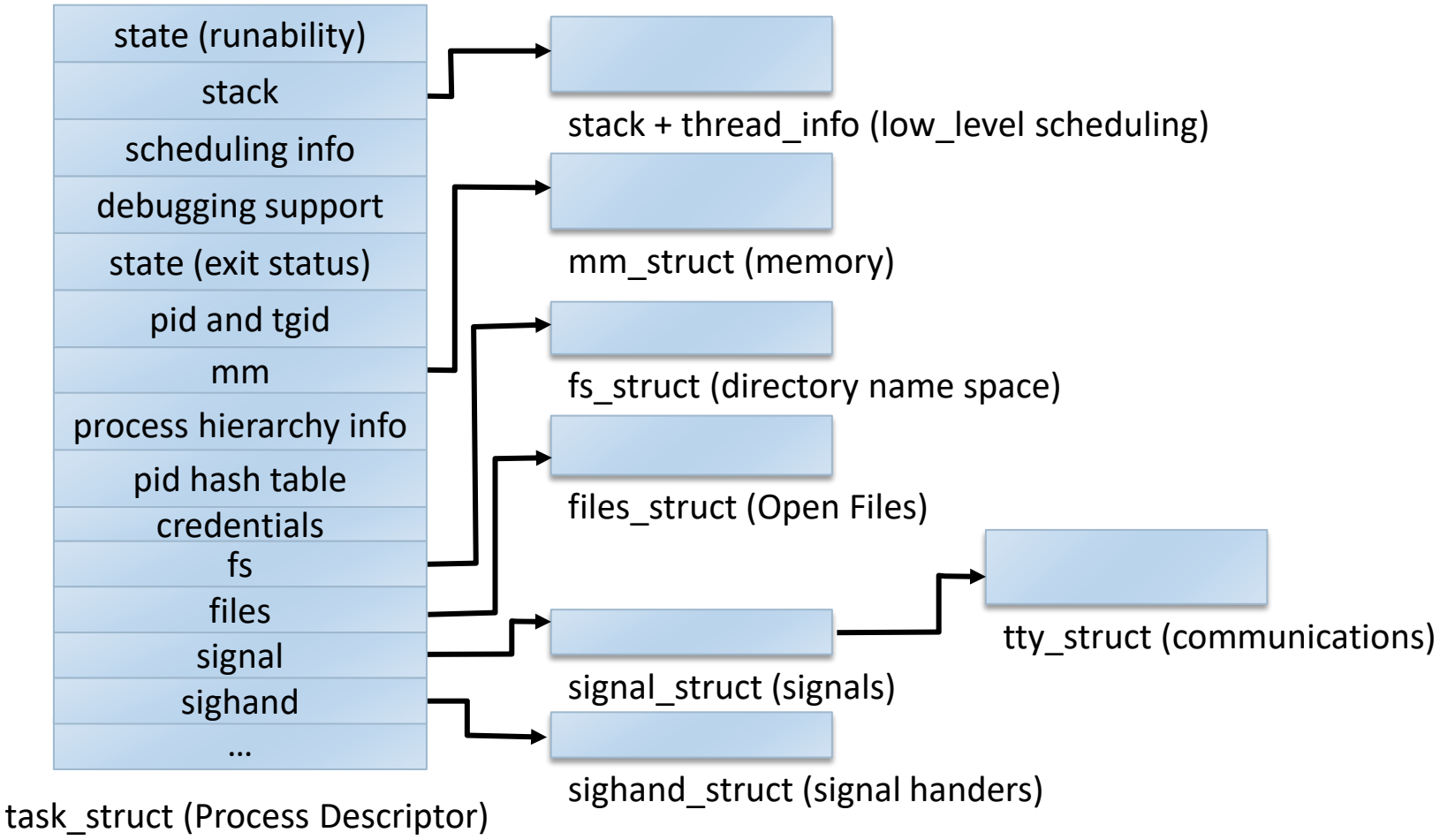
Id.
- Identificador del proceso
- Proceso padre
- Grupo de proceso

control
- Prioridad
- Parámetros del planificador
- Señales
- Instante inicio de ejecución
- Tiempo de uso de CPU
- Tiempo hasta siguiente alarma



- *Process Identification* (PID)
  - Identificador de cara a los usuarios
  - Número positivo de 16 bits (32767) dinámicamente asignado, reusado no de forma inmediata
- *Address of process descriptor* (APD)
  - Identificación dentro del kernel
  - Existe mecanismo PID -> APD (Ej.: hash)

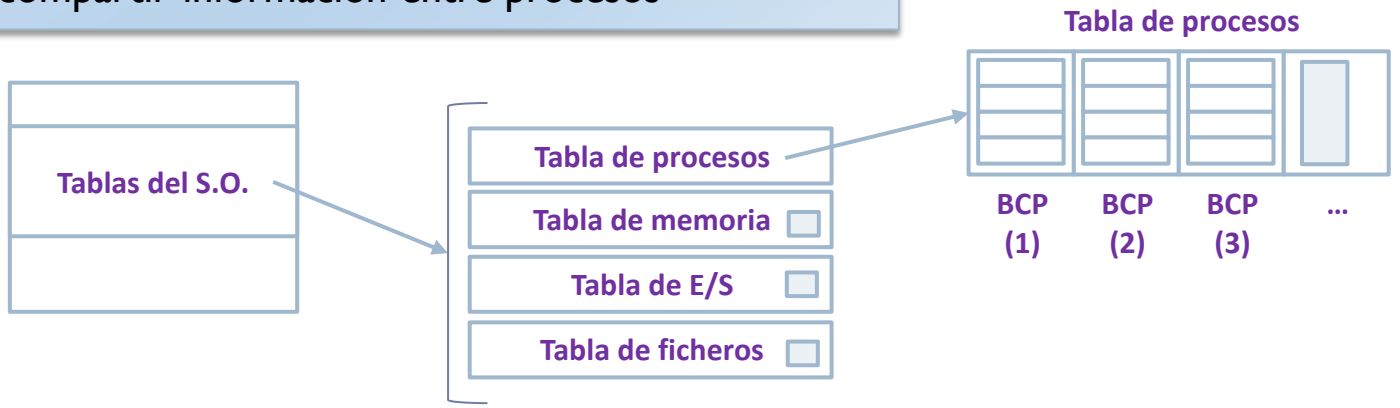
Información del proceso
Linux



Understanding the Linux Kernel (O'Really)
http://www.eecs.harvard.edu/~cs161/assignments/sched.h.html

Información para un proceso
datos gestionados por el S.O.

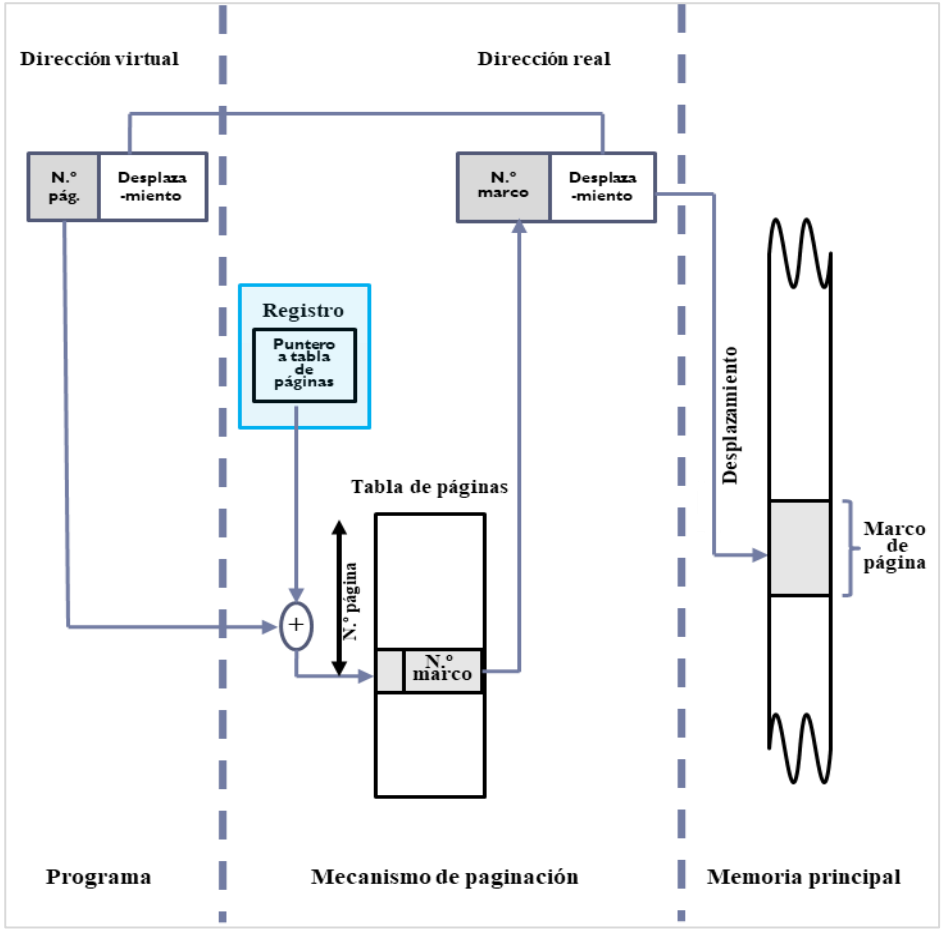
- La información de cada proceso está en el BCP…
- …Información fuera del BCP:
  - Por razones de eficiencia
  - Para compartir información entre procesos



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

Información para un proceso

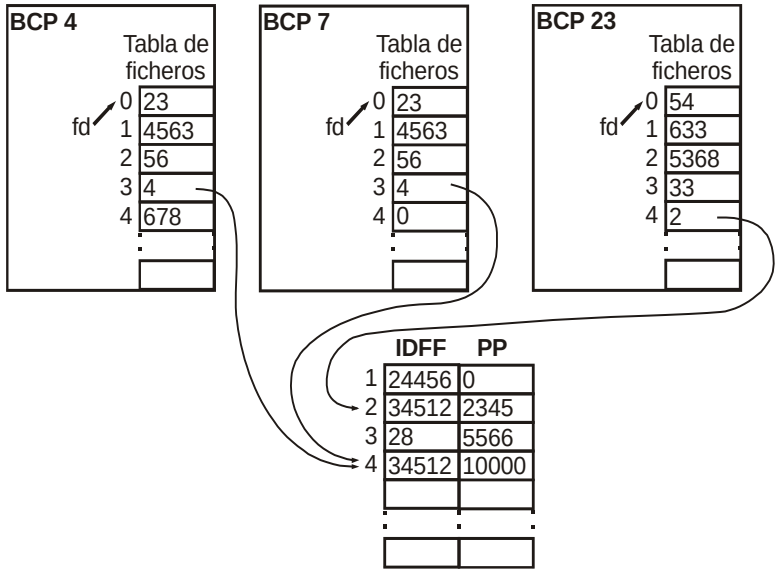
datos gestionados por el S.O. => fuera del BCP



- 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.

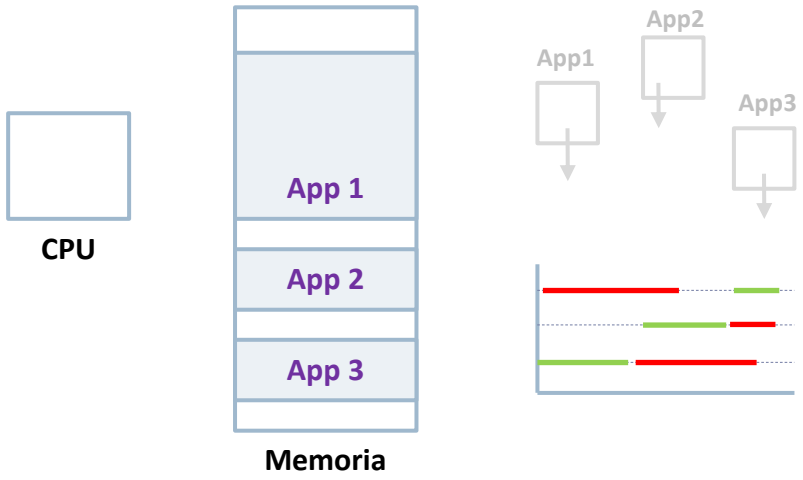
Contenidos

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.
   - Estructura de datos de procesos e hilos en el núcleo
   - Diseño e implementación de la multiprogramación y la multitarea en el núcleo

Modelo ofrecido

repaso

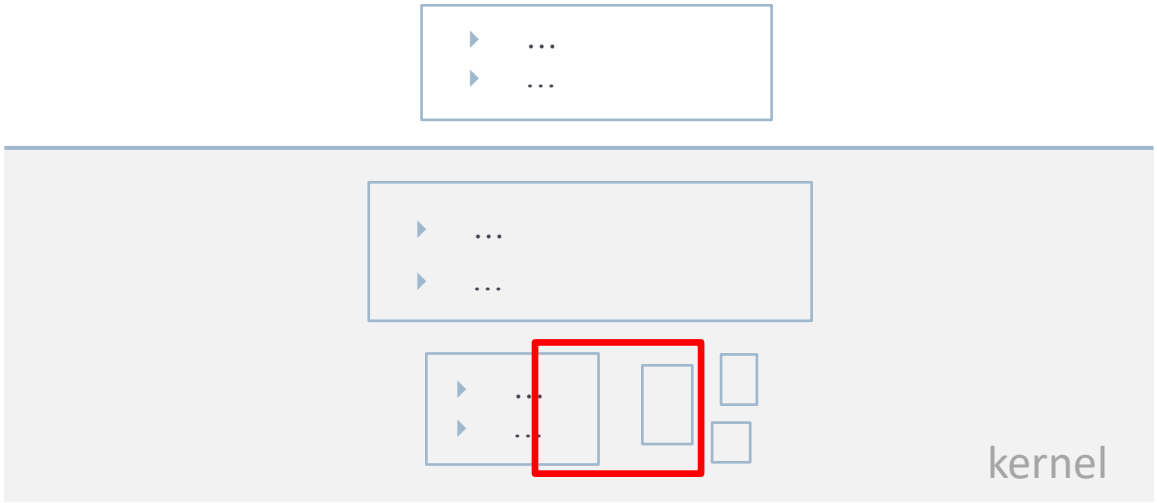
- recursos
- **multiprogramación**
  - protección/compartición
  - jerarquía de procesos
- multitarea
- multiproceso



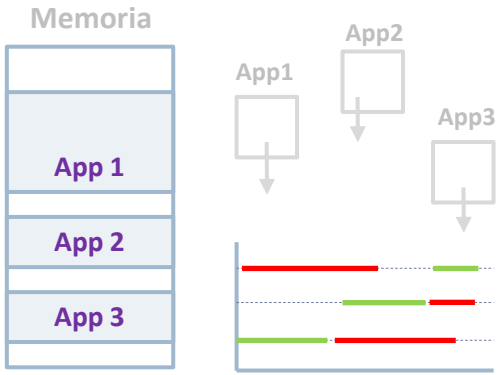
- Multiprogramación
  - Tener varias aplicaciones en memoria
  - Si una aplicación se bloquea por E/S, entonces se ejecuta mientras otra hasta que quede bloqueada
    - Cambio de contexto voluntario (C.C.V.)
  - Eficiencia en el uso del procesador
  - Grado de multiprogramación = número de aplicaciones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

Multiprogramación (datos y funciones)

Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Multiprogramación	• Estado de ejecución • Contexto: registros de CPU… • Lista de procesos	• Int. hw/sw de dispositivos • Planificador • Crear/Destruir/Planificar proceso



Multiprogramación

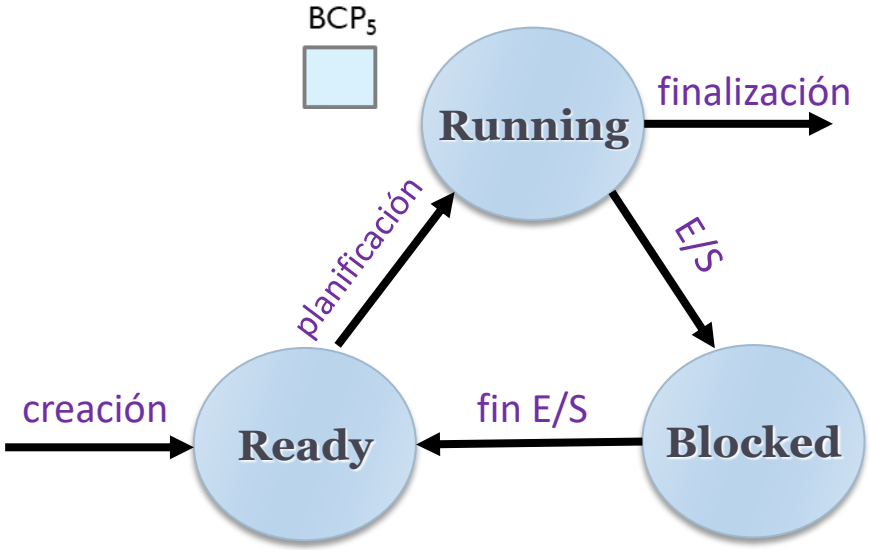


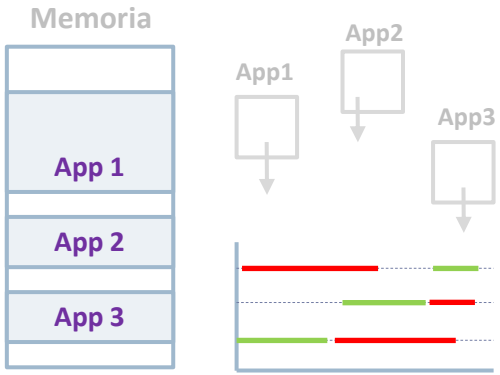
- Tener varias aplicaciones en memoria
- Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - Cambio de contexto voluntario (C.C.V.)

Multiprogramación (datos)

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

- Estado
- Lista/Cola
- Contexto

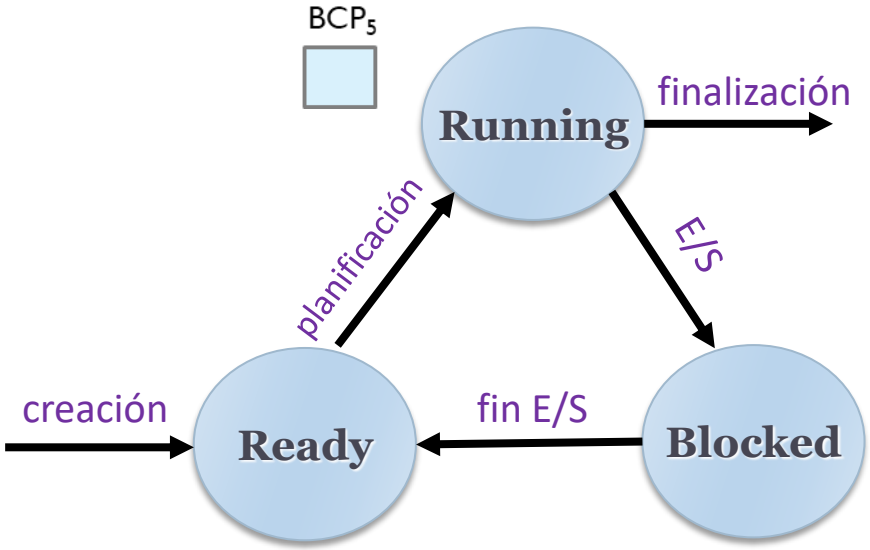




- Tener varias aplicaciones en memoria
- Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - Cambio de contexto voluntario (C.C.V.)

Multiprogramación (datos)
Estados de un proceso (c.c.v.)

- Estado
- Lista/Cola
- Contexto

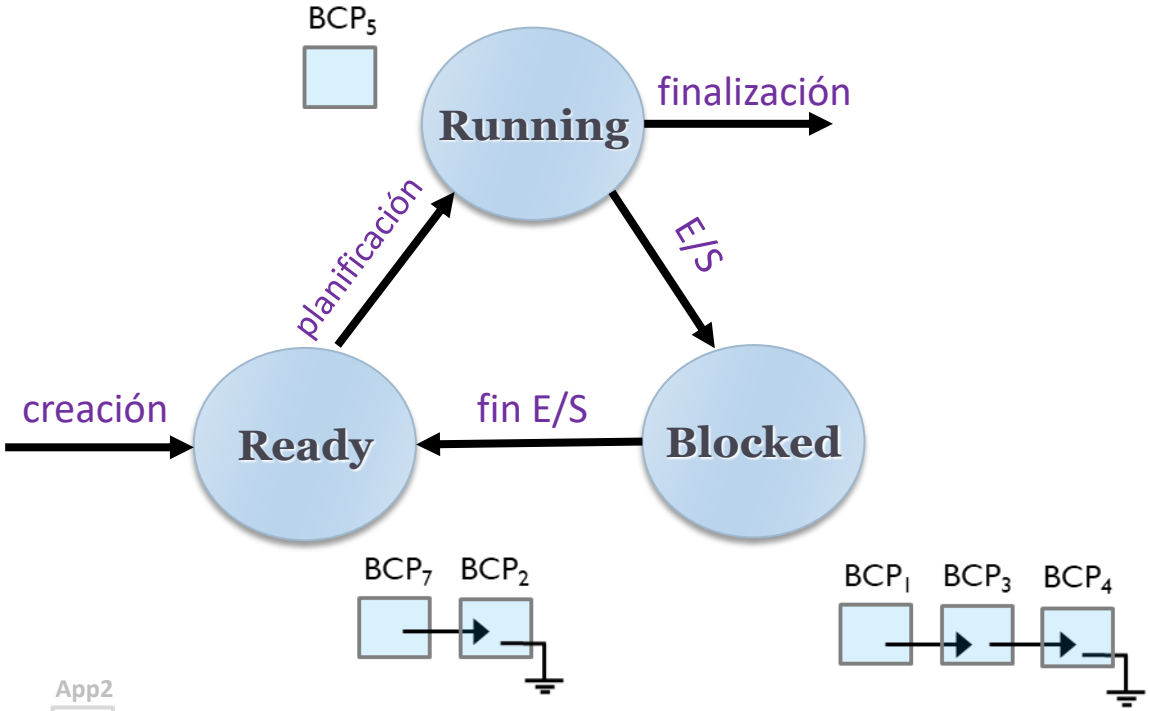


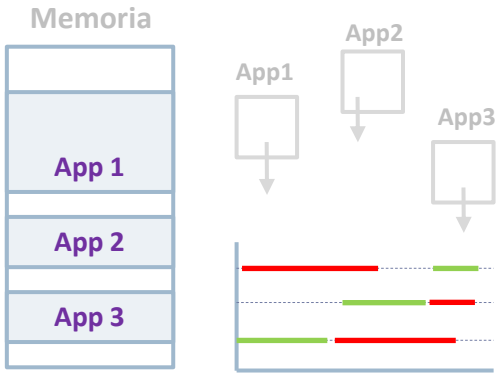
- En ejecución: con CPU asignada
- Listo para ejecutar: no procesador disponible para él
- Bloqueado: esperando un evento
- Suspendido y listo: expulsado pero listo para ejecutar
- Suspendido y bloqueado: expulsado y esperando evento

Multiprogramación (datos)

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

- Estado
- **Lista/Cola**
- Contexto

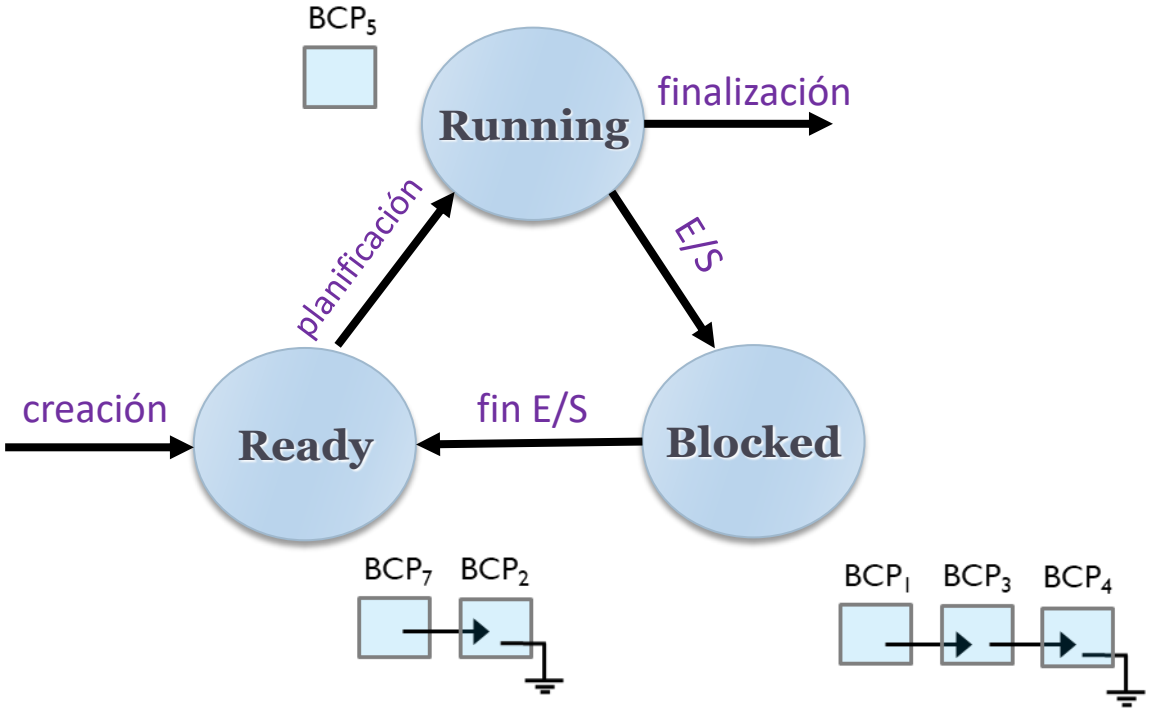




- Tener varias aplicaciones en memoria
- Si una aplicación se bloquea por E/S, entonces se ejecuta otra (hasta que quede bloqueada)
  - Cambio de contexto voluntario (C.C.V.)

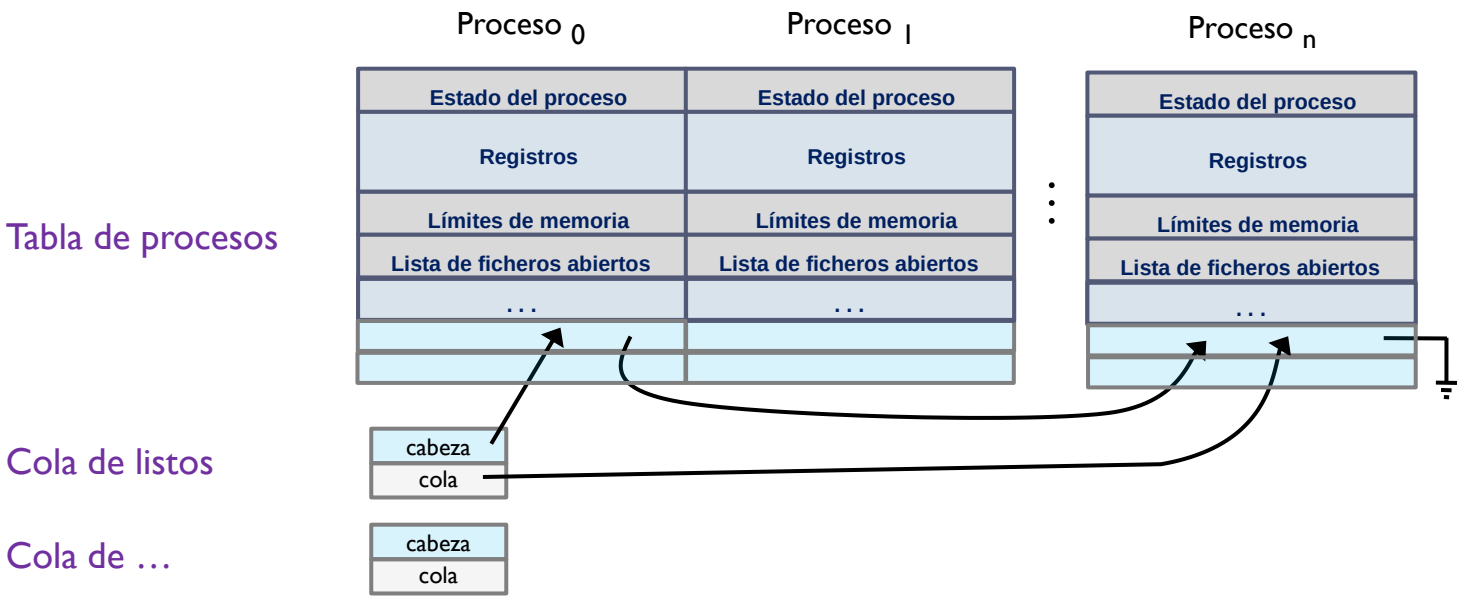
Multiprogramación (datos)
Lista/Colas de procesos (c.c.v.)

- 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

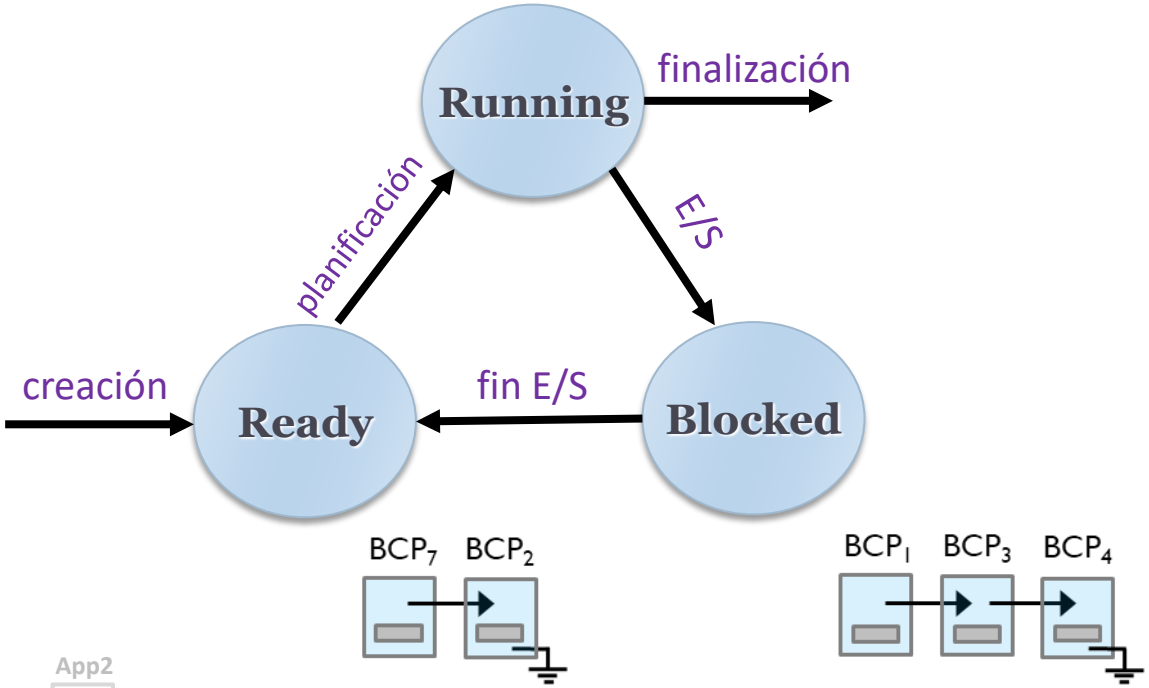


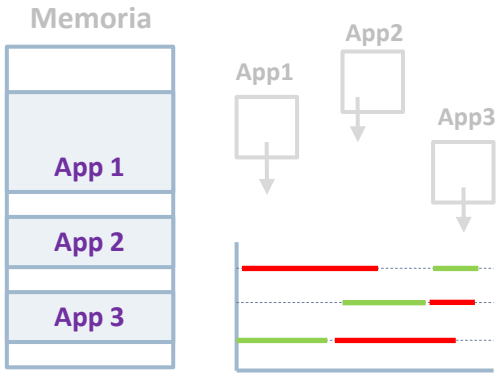
- Cola de listos: procesos esperando a ejecutar en CPU
- Cola de bloqueados por recurso: procesos a la espera de finalizar una petición bloqueante al recurso asociado
  - Un proceso solo puede estar en una cola (como mucho)

Multiprogramación (datos)

Contexto de un proceso

- Estado
- Lista/Cola
- **Contexto**



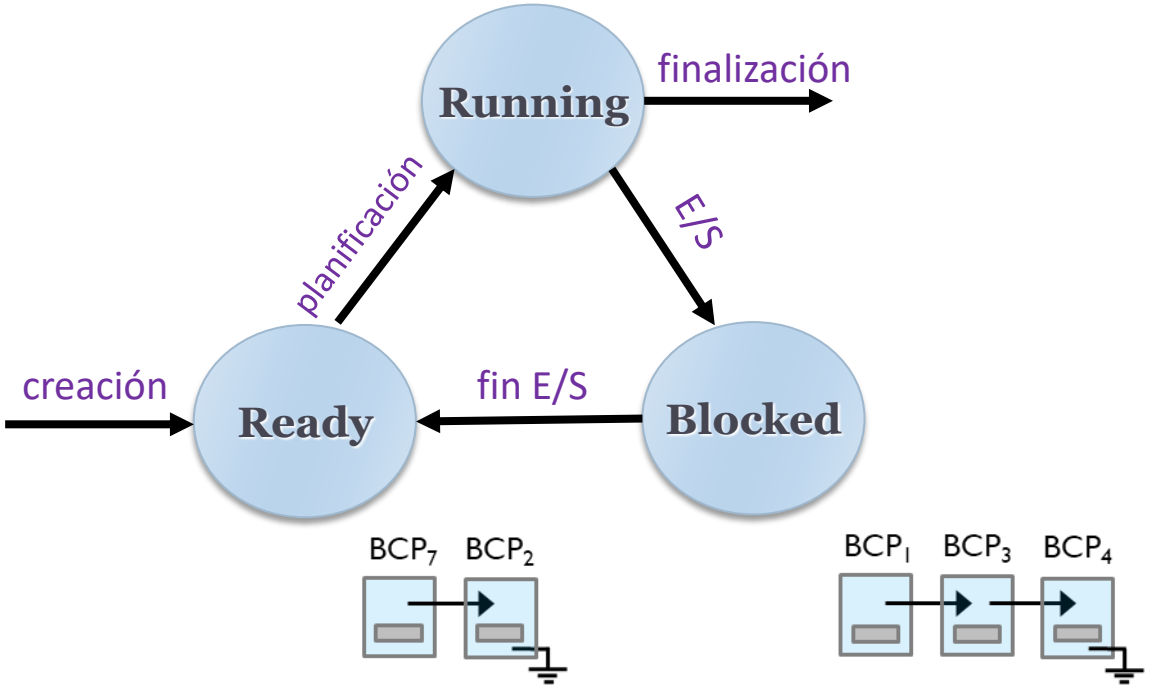


- Tener varias aplicaciones en memoria
- **Cambio de contexto**:
  - Si proceso se bloquea entonces SO asigna CPU a otro proceso (hasta que bloqueado/terminado)

Multiprogramación (datos)

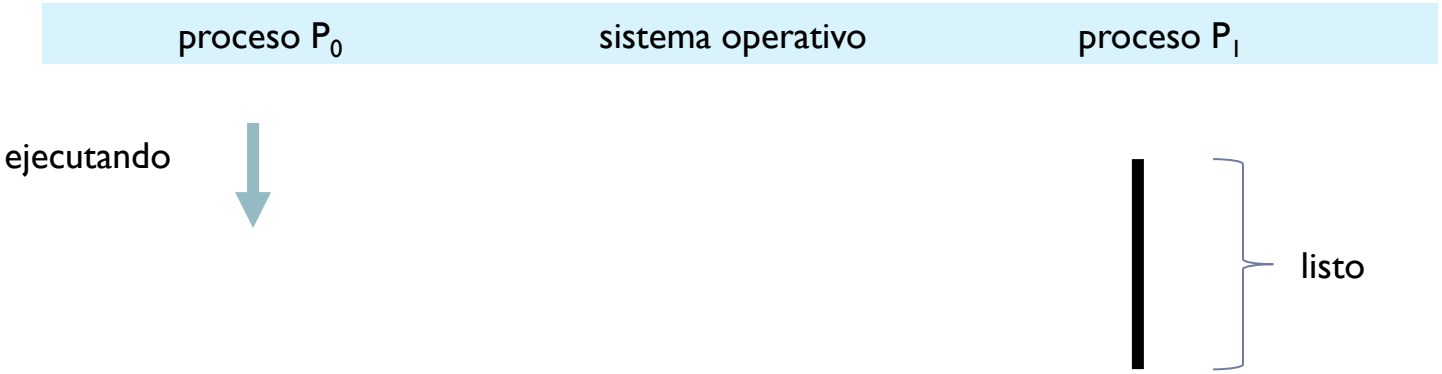
Contexto de un proceso

- Estado
- Lista/Cola
- **Contexto**

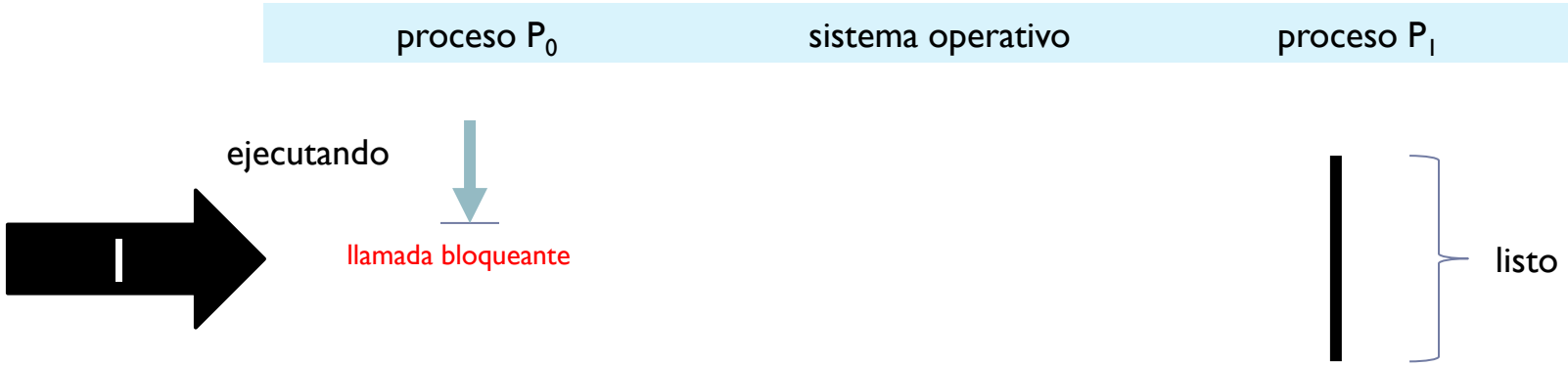


- Registros generales: PC, RE, etc.
- Registros específicos: Registros de coma flotante, etc.
- Referencias a recursos: puntero a código, datos, etc.

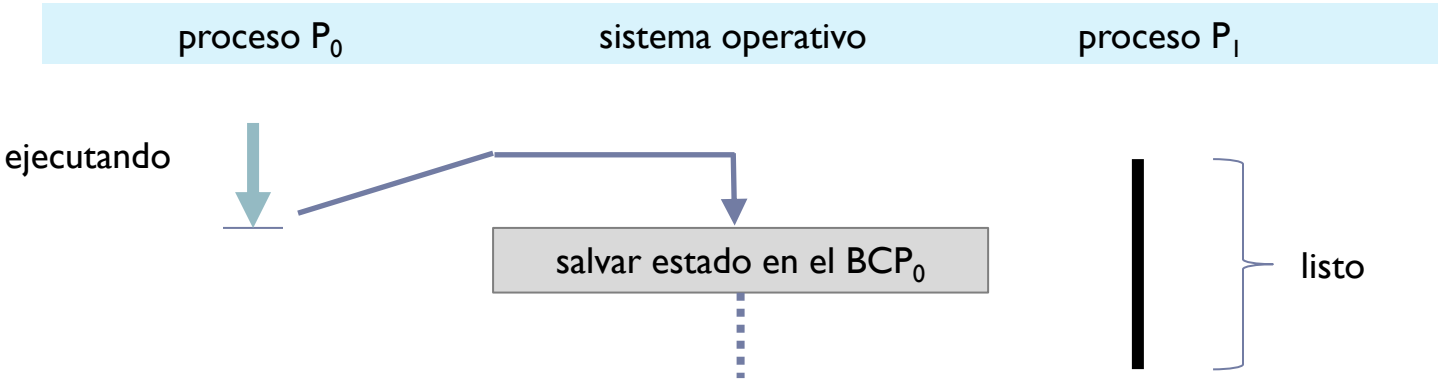
Multiprogramación: cambios de contexto



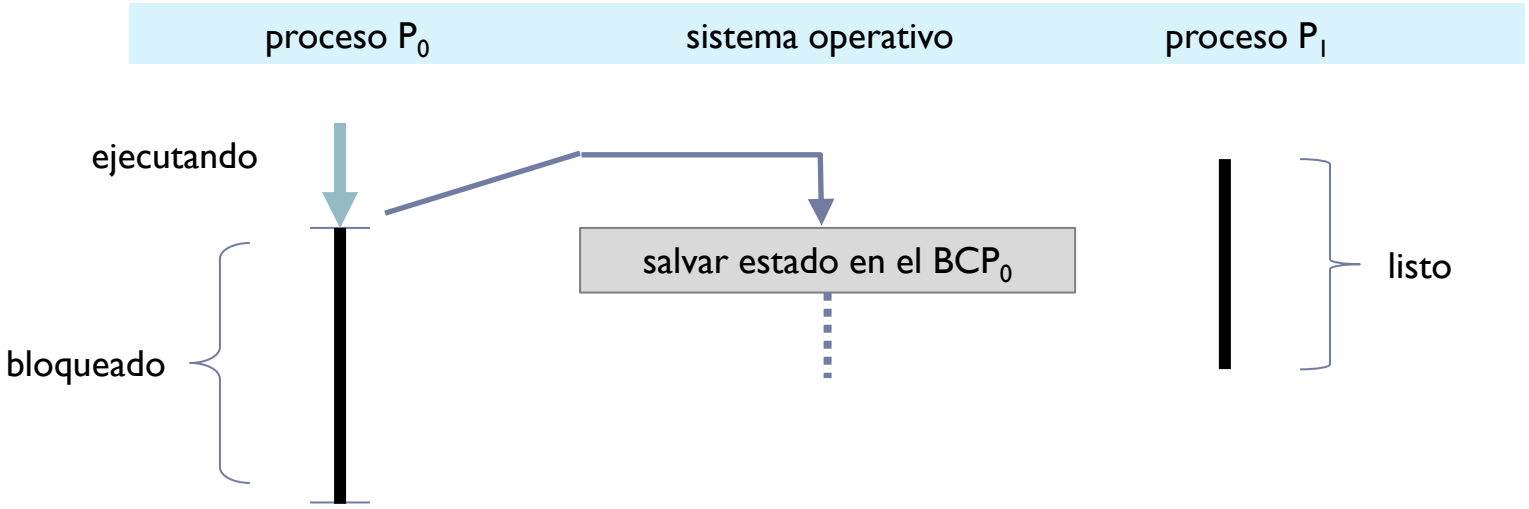
Multiprogramación: cambios de contexto



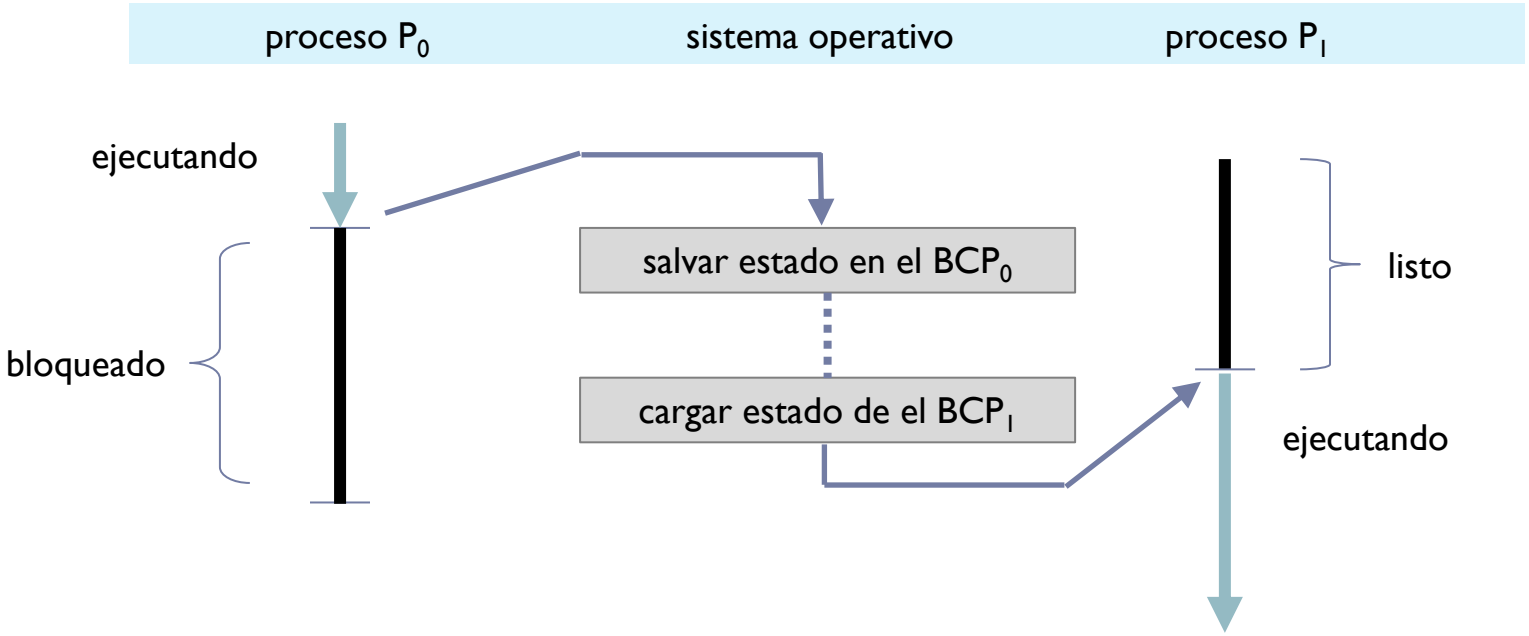
Multiprogramación: cambios de contexto



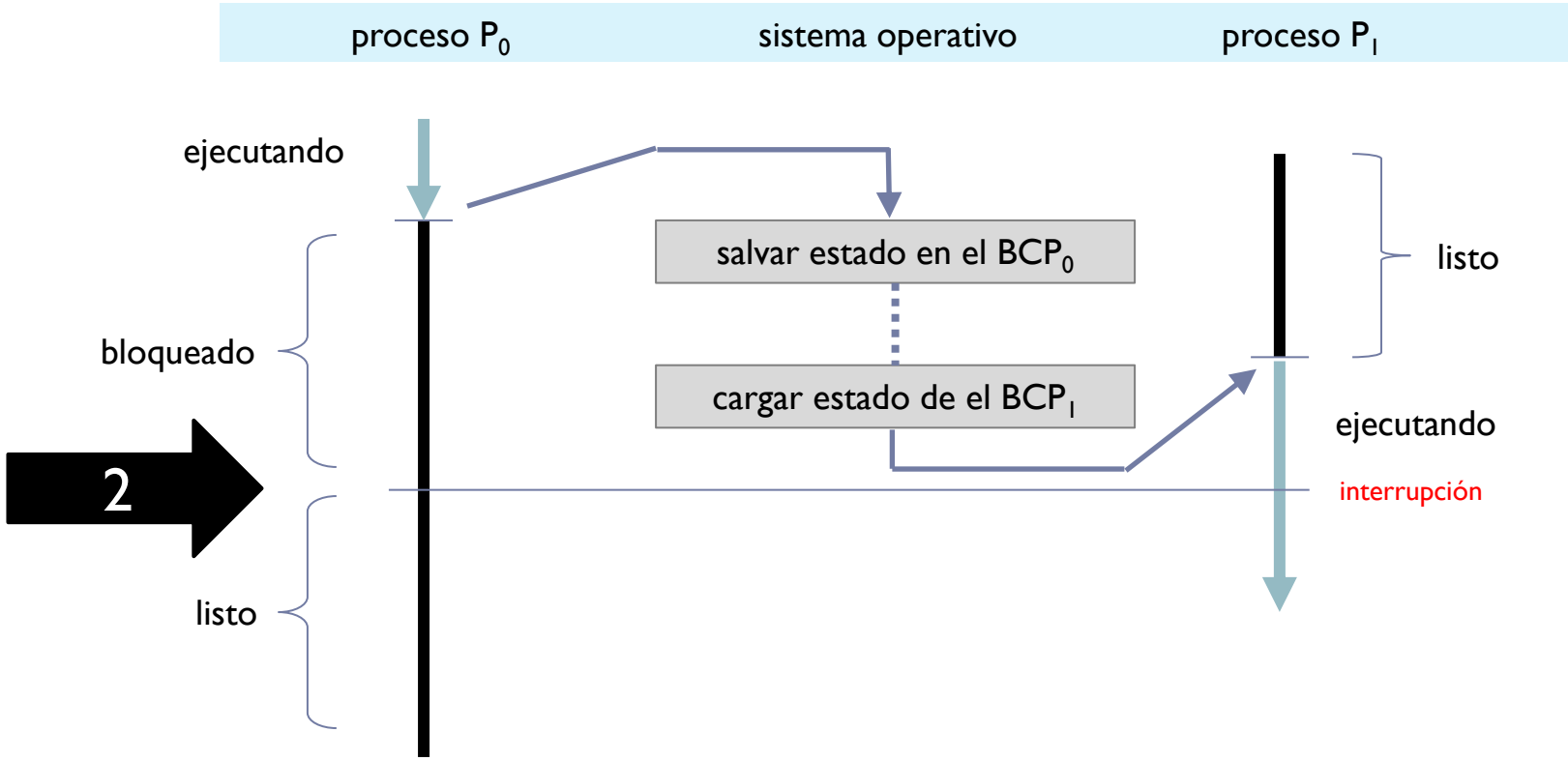
Multiprogramación: cambios de contexto



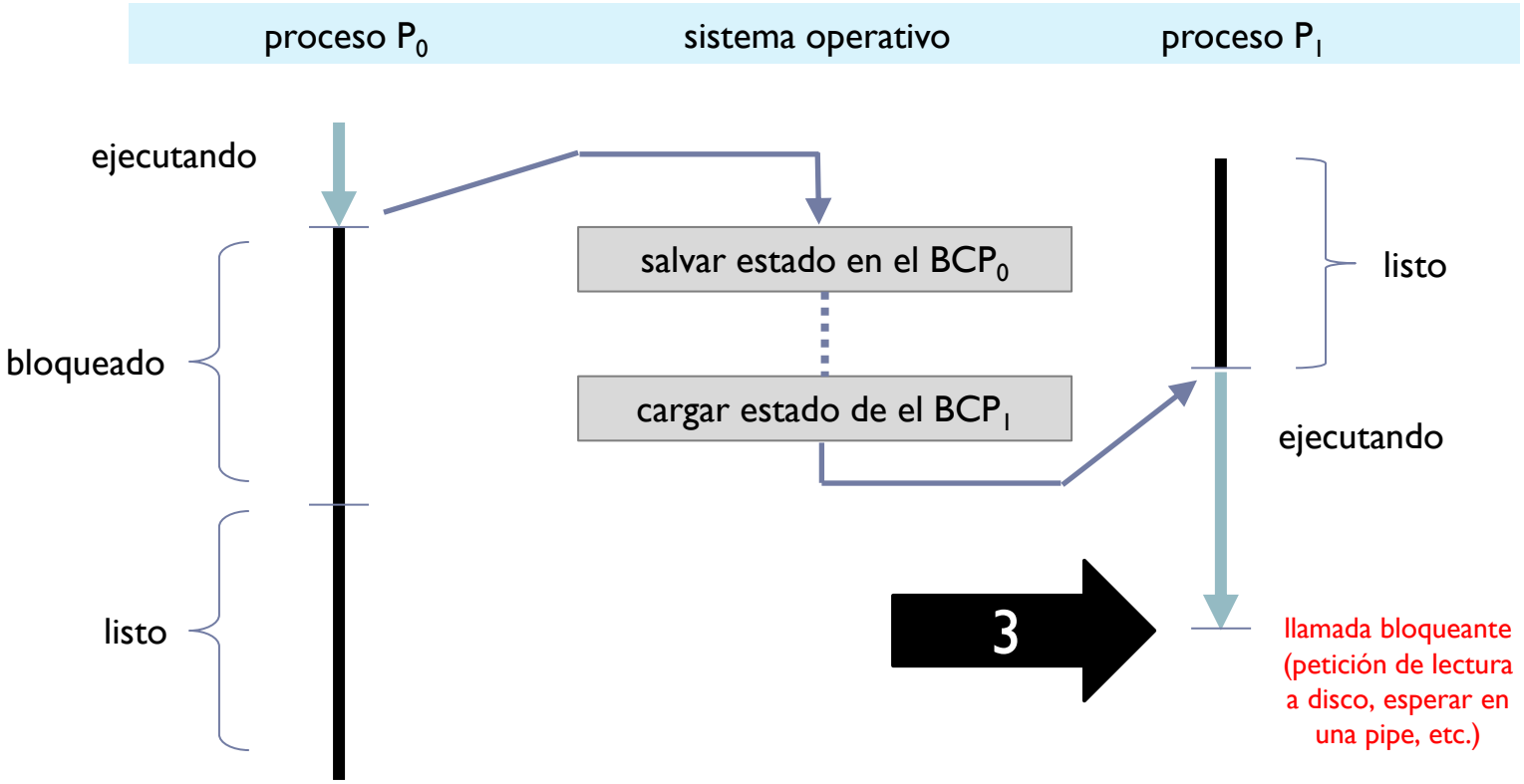
Multiprogramación: cambios de contexto



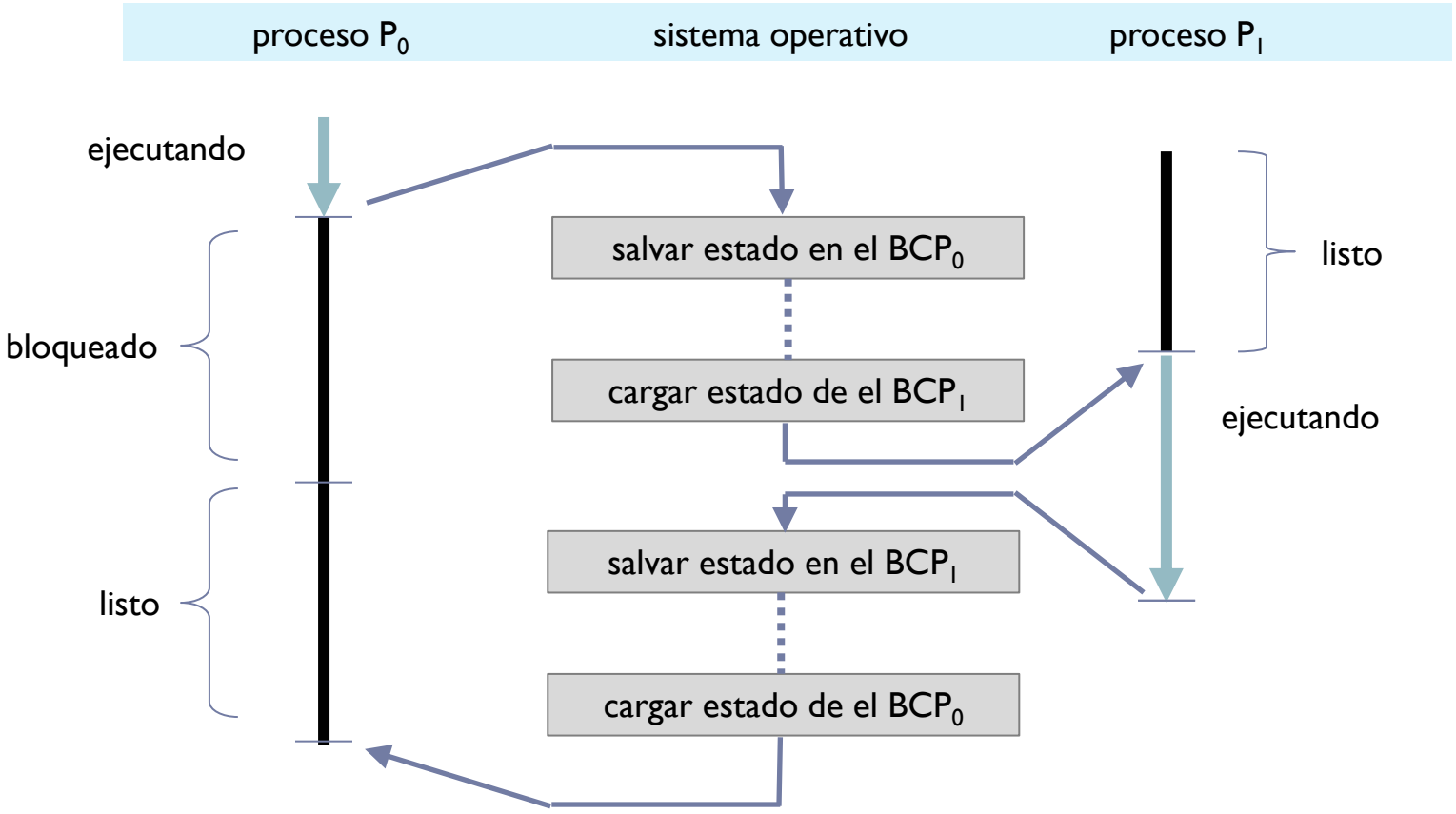
Multiprogramación: cambios de contexto



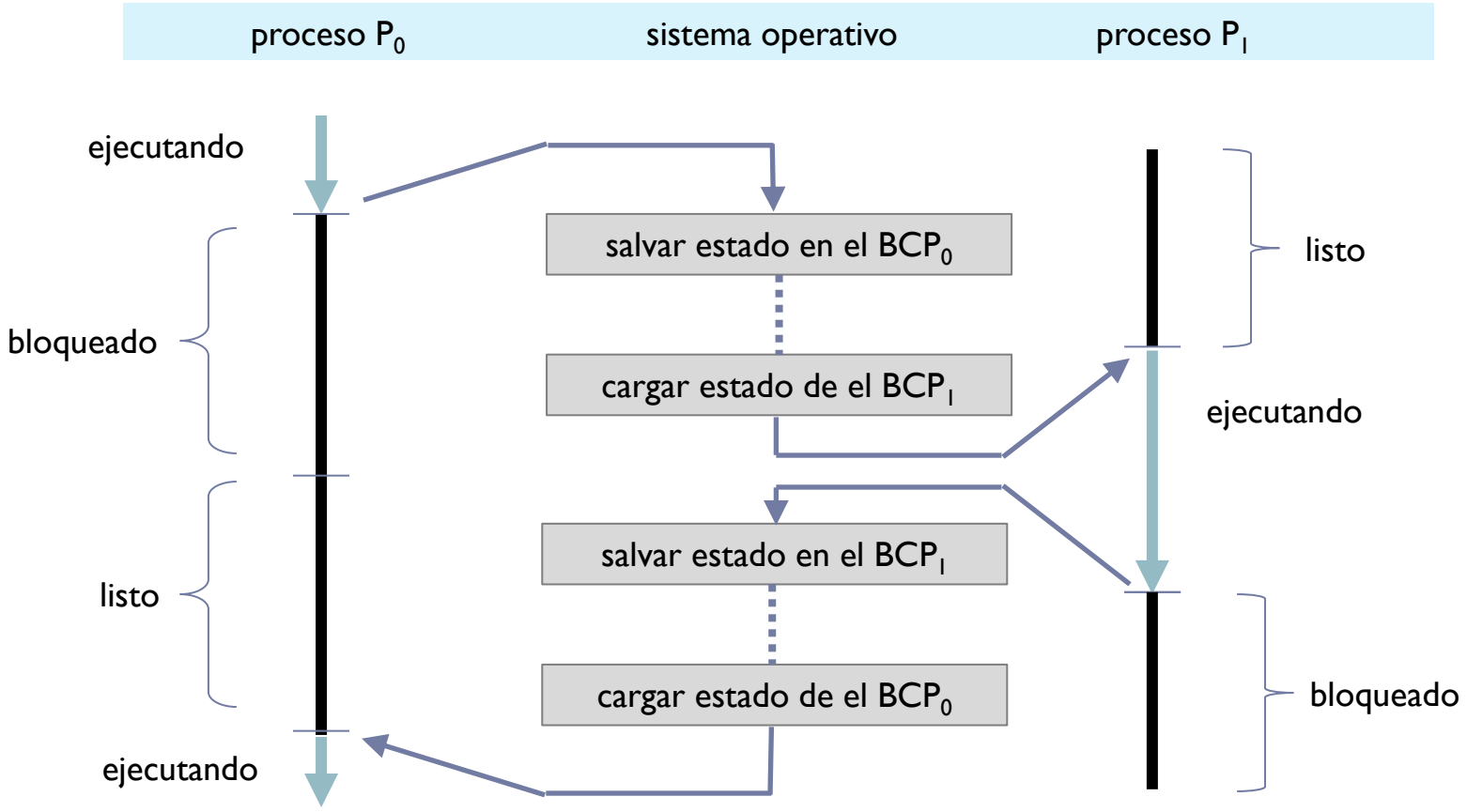
Multiprogramación: cambios de contexto



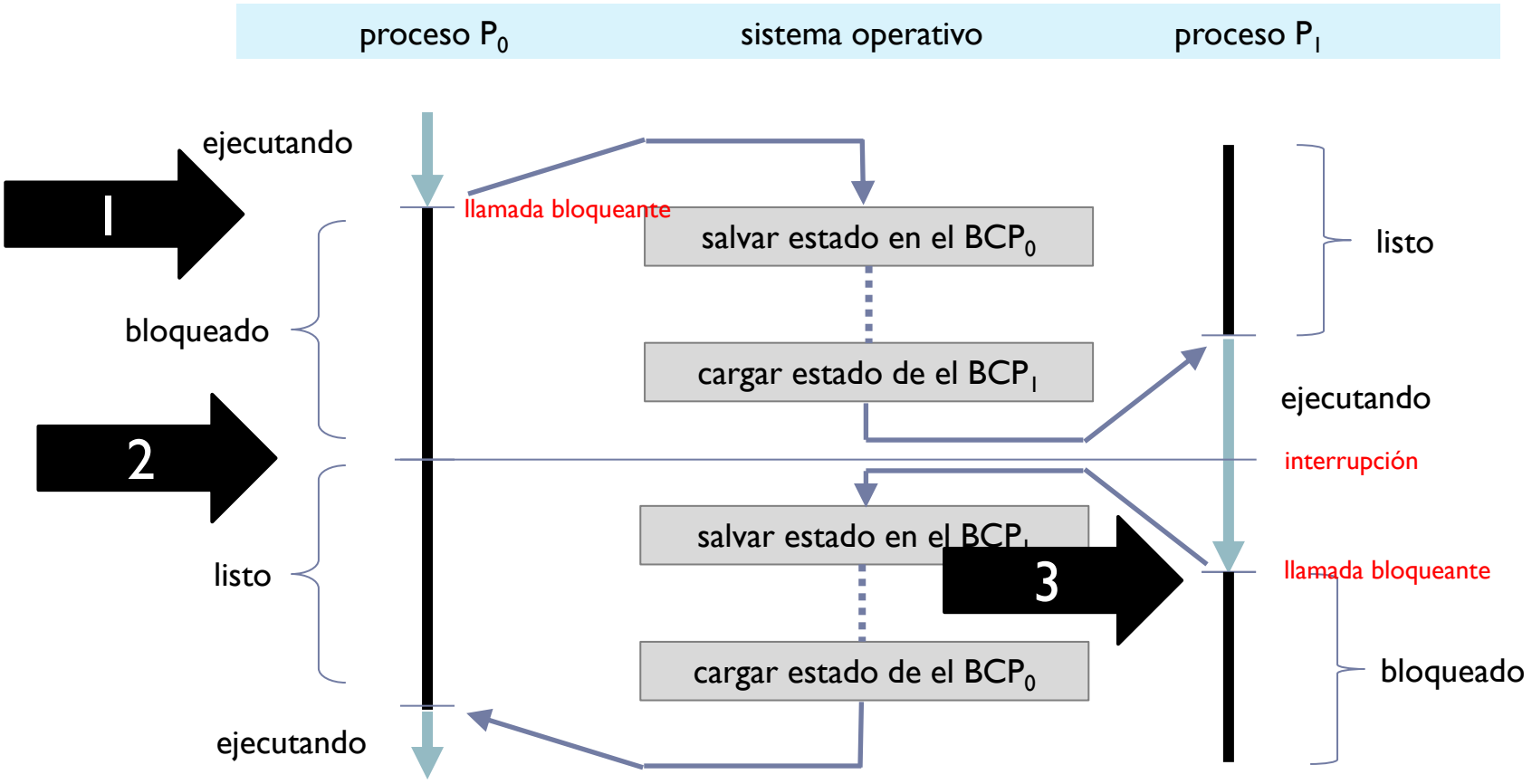
Multiprogramación: cambios de contexto



Multiprogramación: cambios de contexto



Multiprogramación: cambios de contexto



Pseudocódigo de ejemplo (P0)

planificador()

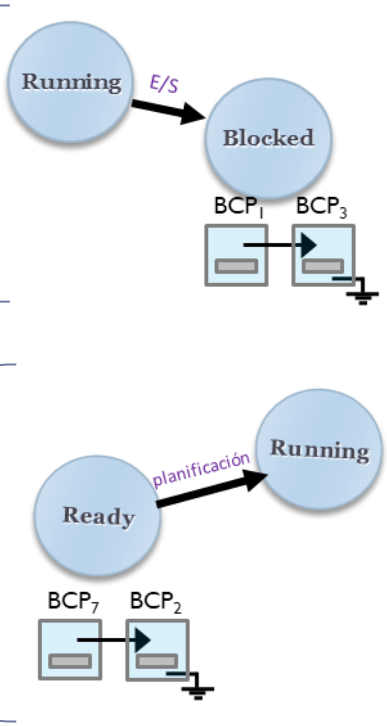
- return extraer(CPU_Listos);

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_0

cargar estado en BCP_1



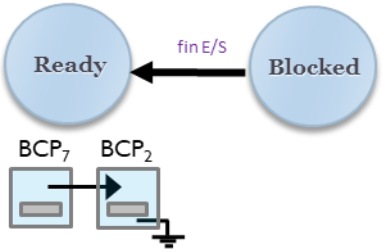
Pseudocódigo de ejemplo (P1)

2

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;



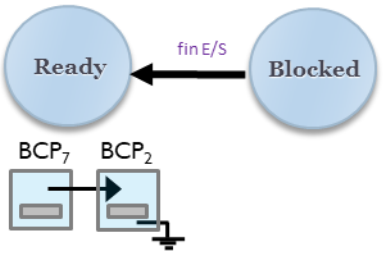
Pseudocódigo de ejemplo (P1)

2

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)

3

planificador()

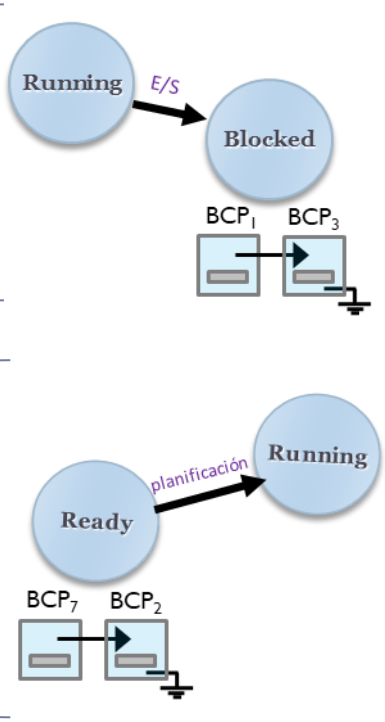
- return extraer(CPU_Listos);

Disco_LeerBloqueDisco()

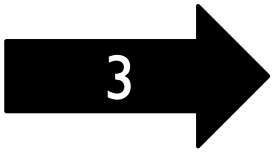
- Si (no hay bloque en caché)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Disco_Procesos, procesoActual);
  - proceso = procesoActual;
- procesoActual = planificador();
- procesoActual->estado = EJECUCION;
- cambio_contexto(&(proceso->contexto), &(procesoActual->contexto));
- return extraer(Disco_caché, bloque) ;

salvar estado en BCP_1

cargar estado en BCP_0



Pseudocódigo de ejemplo (P0)



Disco_LeerBloqueDisco()

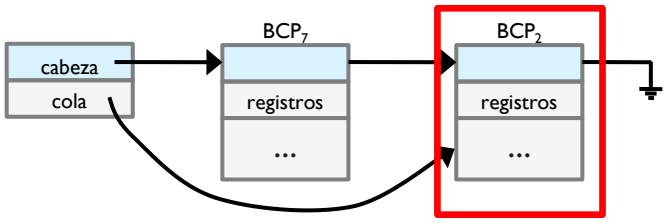
- Si (no hay bloque en caché)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Disco_Procesos, procesoActual);
  - proceso = procesoActual;
  - procesoActual = planificador();
  - procesoActual->estado = EJECUCION;
  - cambio_contexto(&(proceso->contexto), &(procesoActual->contexto));
- return extraer(Disco_caché, bloque) ;

Teclado_LeerTecla()

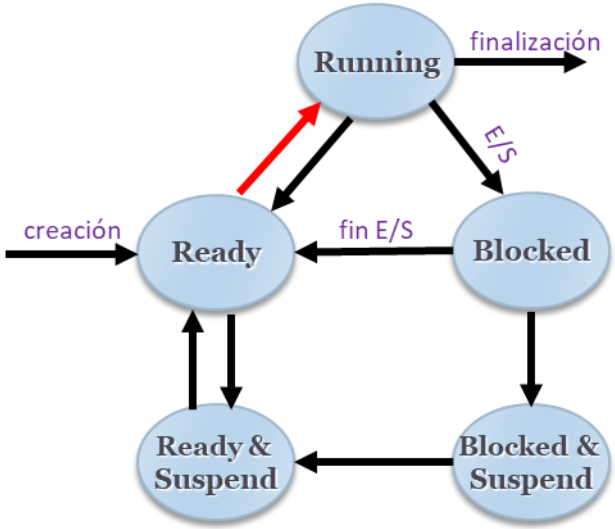
- Si (no hay tecla)
  - procesoActual->estado = BLOQUEADO;
  - Insertar(Teclado_Procesos, procesoActual);
  - proceso = procesoActual;
  - procesoActual = planificador();
  - procesoActual->estado = EJECUCION;
  - cambio_contexto(&(proceso->contexto), &(procesoActual->contexto));
- return extraer(Teclado_Teclas) ;

Planificador y activador

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



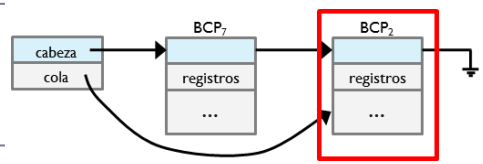
- Activador:
  Da control al proceso que el planificador ha seleccionado (cambio de contexto - restaurar)

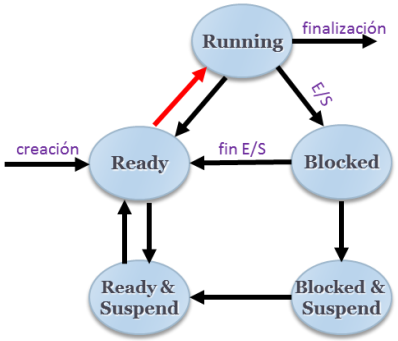


Planificador y activador

planificador()

- return extraer(CPU_Listos);





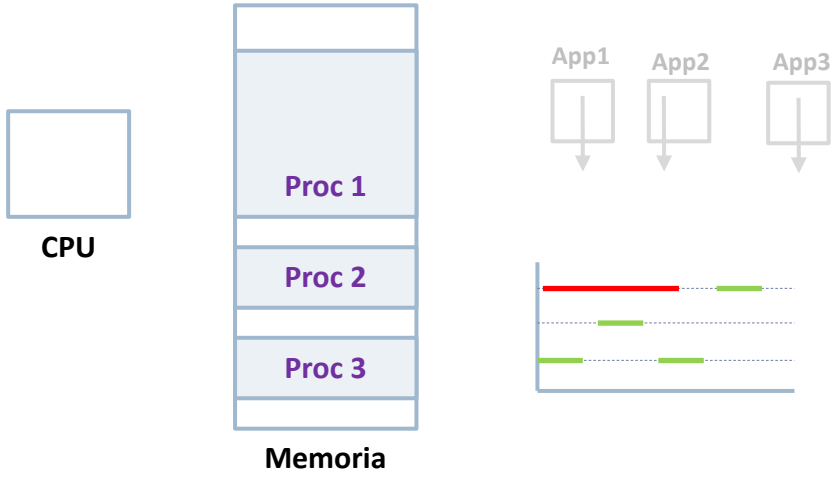
Teclado_LeerTecla()

- 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)

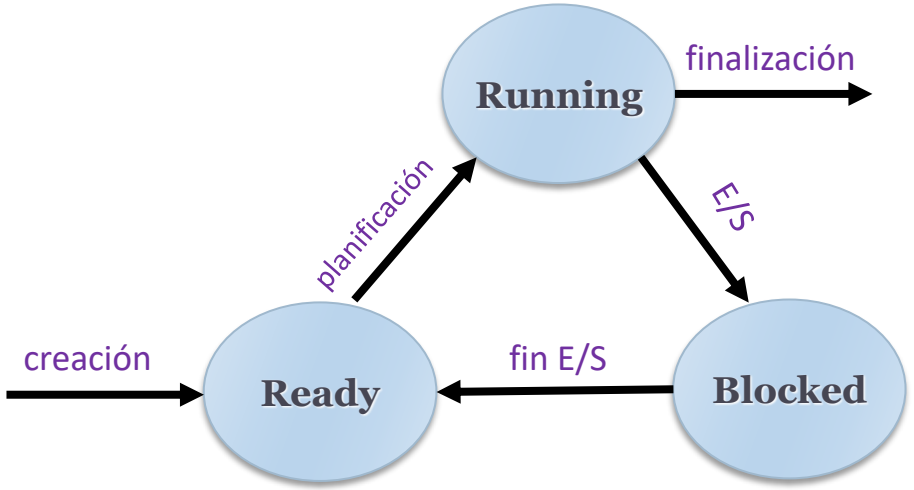
Requisitos	Información (en estructuras de datos)	Funciones (internas, servicio y API)
Recursos	• Zonas de memoria (código, datos y pila) • Archivos abiertos • Señales activas	• Diversas funciones internas • Diversas funciones de servicio para memoria, ficheros, etc.
Multiprogramación	• Estado de ejecución • Contexto: registros de CPU… • Lista de procesos	• Int. hw/sw de dispositivos • Planificador • Crear/Destruir/Planificar proceso

○ Protección / Compartición	• Paso de mensajes
	  • Cola de mensajes de recepción
	• Memoria compartida
	  • Zonas, locks y conditions
		• Envío/Recepción mensaje y gestión de la cola de mensaje
		• API concurrencia y gestión de estructuras de datos

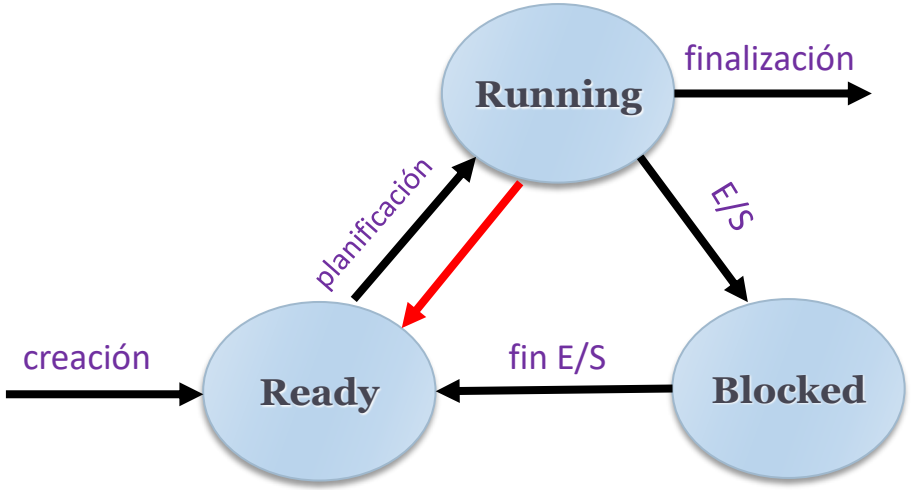
○ Jerarquía de procesos	• Relación de parentesco • Conjuntos de procesos relacionados • Procesos de una misma sesión	• Clonar/Cambiar imagen de proceso • Asociar procesos e indicar proceso representante
Multitarea	• Quantum restante • Prioridad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso
Multiproceso	• Afinidad	• Int. hw/sw de reloj • Planificador • Crear/Destruir/Planificar proceso

Estados de un proceso

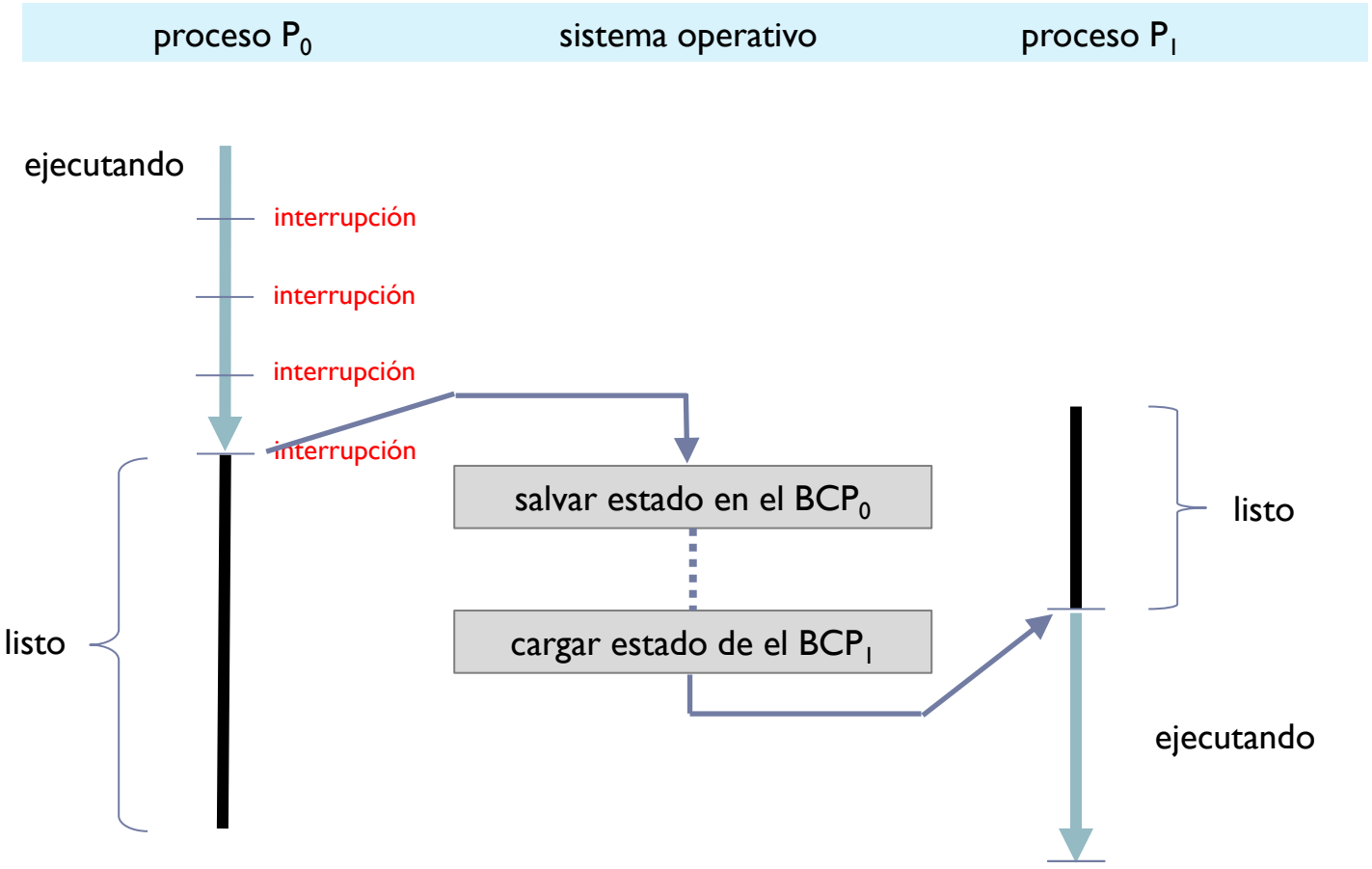
C.C.V.



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



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



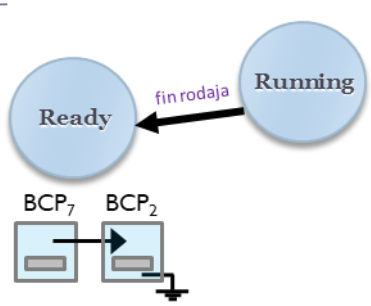
Pseudocódigo de ejemplo (P0)

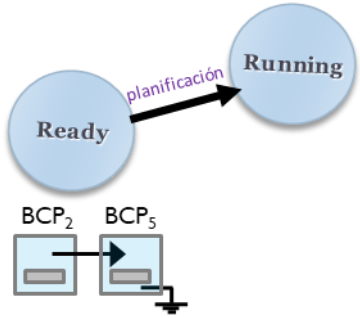
```
Reloj_Interrupción_Hardware ()
  • Ticks++;

  • 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_0

cargar estado en BCP_I





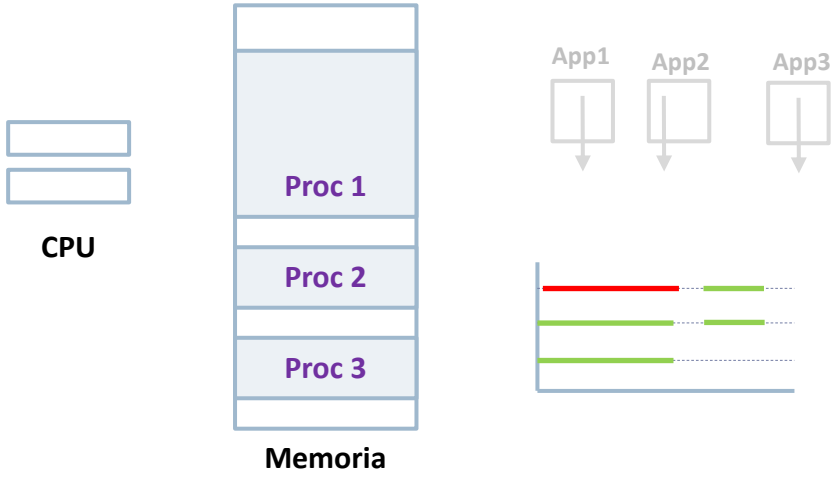
Tipos de cambio de contexto
resumen

- **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

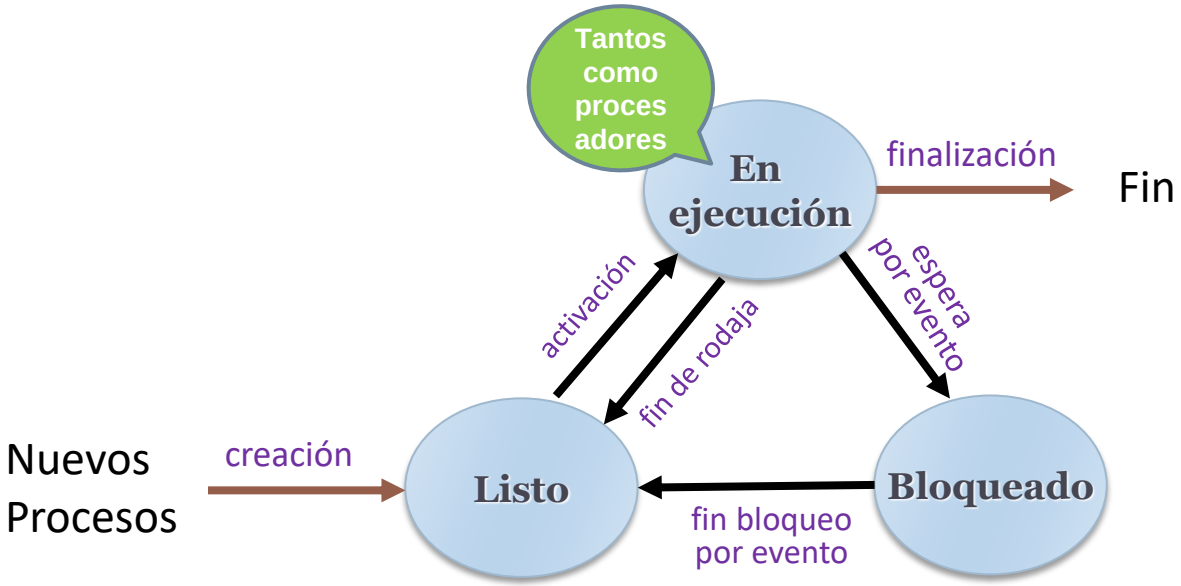
- recursos
- multiprogramación
  - protección/compartición
  - jerarquía de procesos
- multitarea
- **multiproceso**



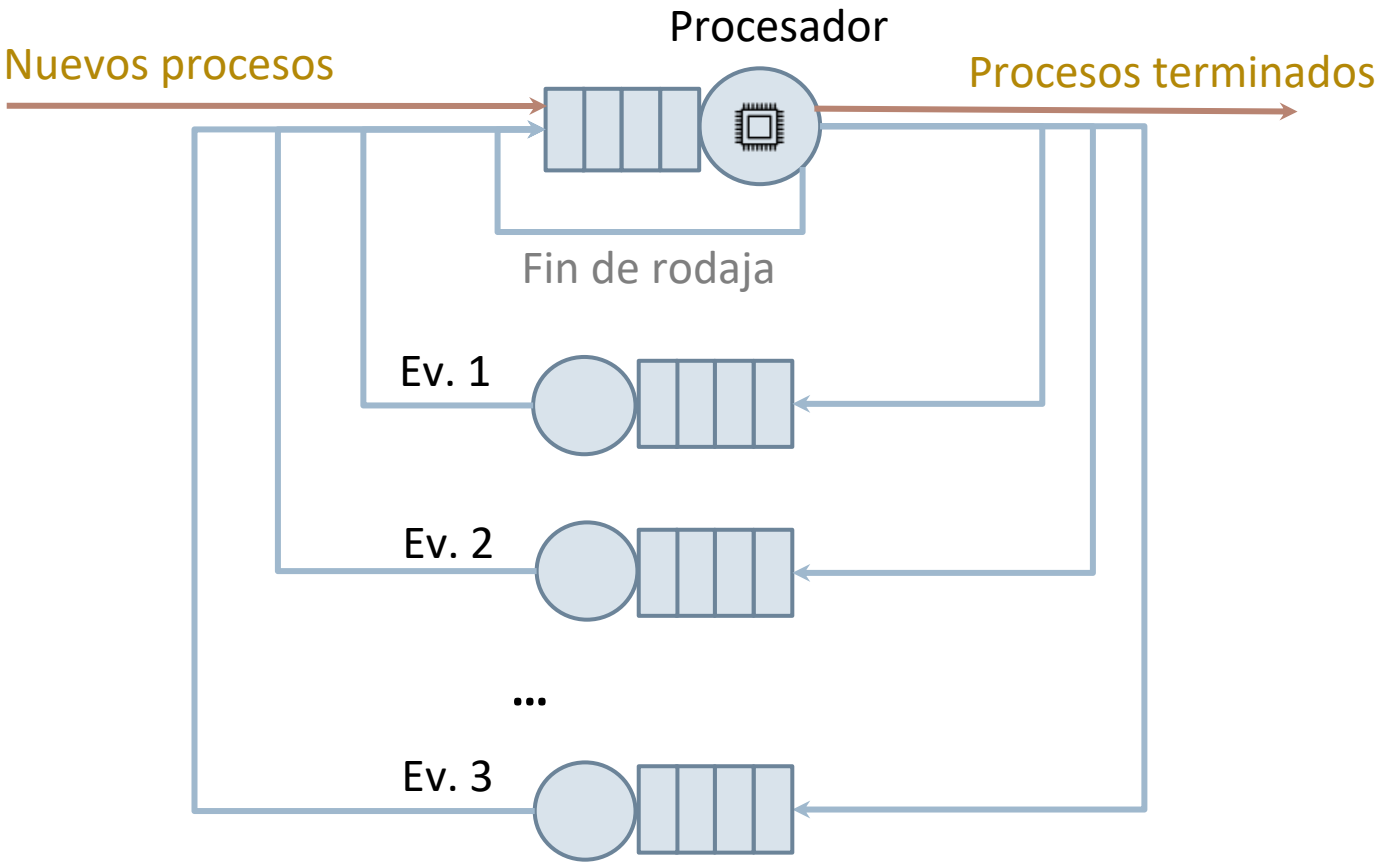
- Multiproceso
  - Se dispone de varios procesadores (multicore/multiprocesador)
  - Además del reparto de cada CPU (multitarea) hay paralelismo real entre varias tareas (tantas como procesadores)
    - Se suele usar planificador y estructuras de datos separadas por procesador con algún mecanismo de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

Ciclo de vida básico de un proceso

- Estado
- Lista/Cola
- Contexto

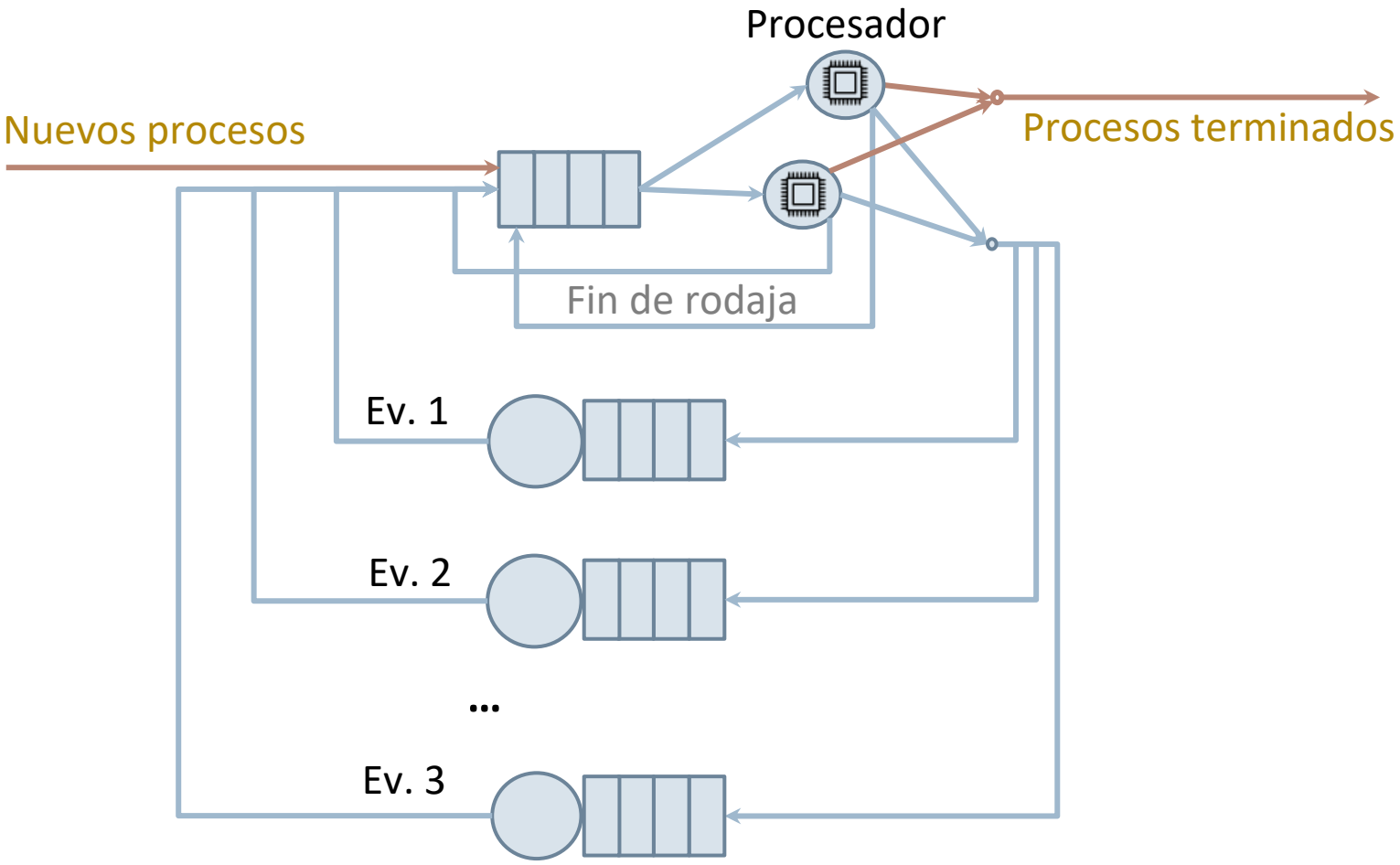


Modelo de colas simplificado: 1 procesador



Sistemas operativos: una visión aplicada (© J. Carrete et al.)
https://iconarchive.com/show/outline-icons-by-iconsmind/CPU-icon.html

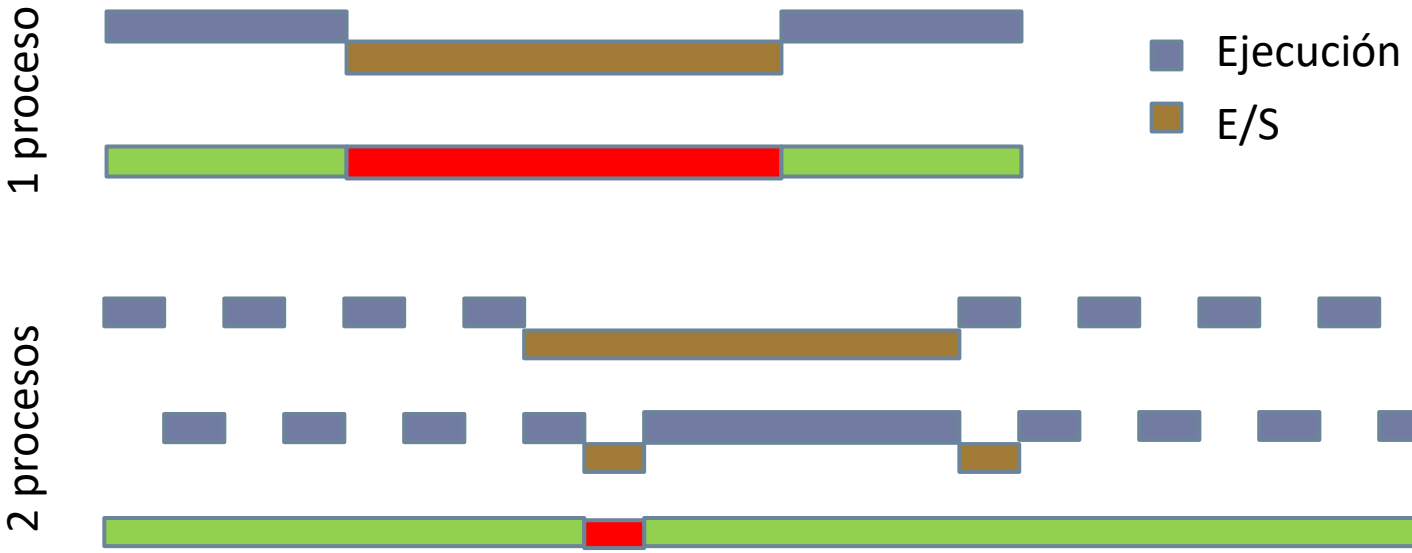
Modelo de colas simplificado: N procesadores



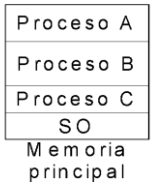
Sistemas operativos: una visión aplicada (© J. Carrete et al.)
https://iconarchive.com/show/outline-icons-by-iconsmind/CPU-icon.html

Ventajas de la multitarea

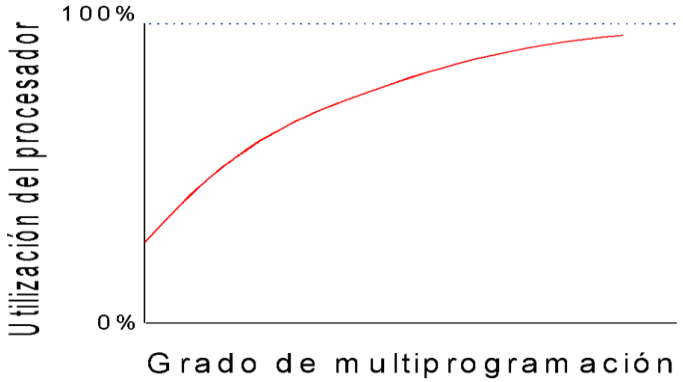
- Modularidad: facilita la programación dividiendo los programas en procesos.
- Permite el servicio interactivo simultáneo de N usuarios de forma eficiente.
- Aprovecha los tiempos que los procesos pasan esperando a que se completen sus operaciones de E/S.
- Aumenta el uso de la CPU.

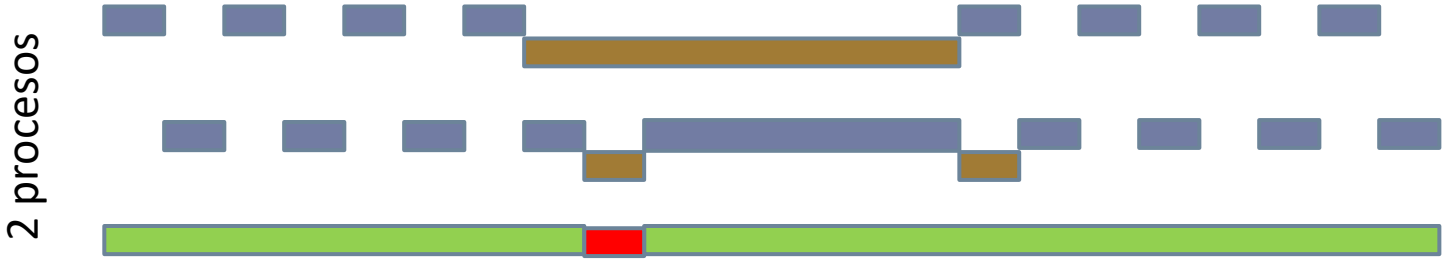


Ventajas de la multitarea

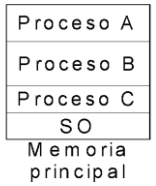


- El uso de la CPU… depende del grado de multiprogramación
- Grado de multiprogramación: nº de procesos activos.
- ¿Siempre más procesos mejora el % de utilización de la CPU?

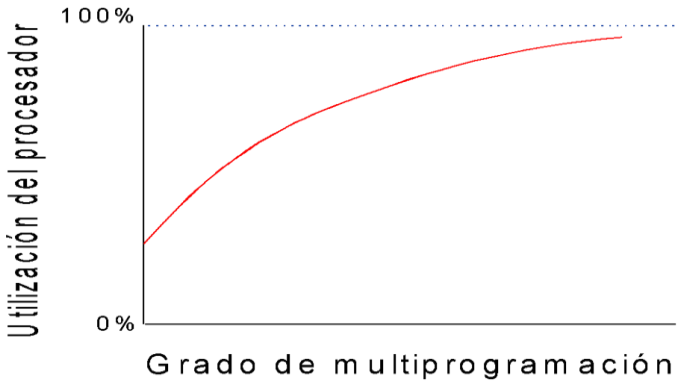




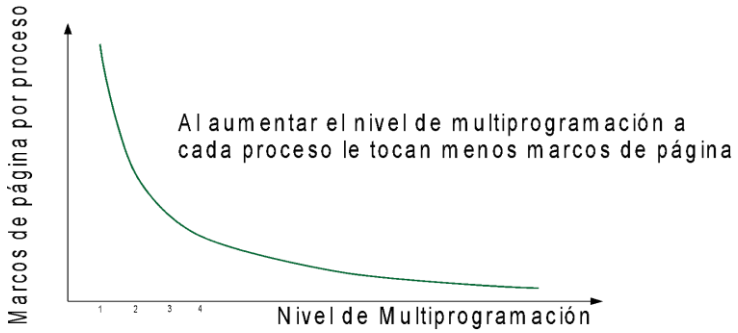
Multiprogramación y memoria



- Los sistemas **sin** memoria virtual:
  - Cada proceso reside totalmente en M.P.

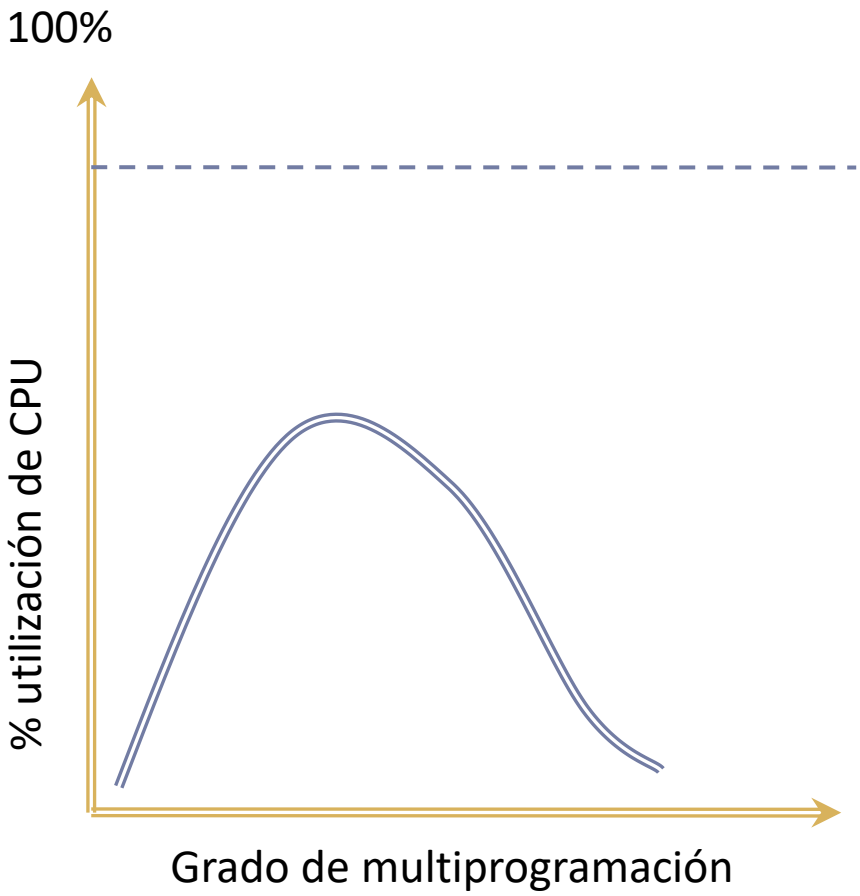


- Los sistemas **con** memoria virtual:
  - Dividen el espacio direccionable de los procesos en páginas.
  - Dividen la memoria física principal en marcos de página.
  - En un momento dado cada proceso tiene un cierto número de sus páginas en memoria principal (**conjunto residente**).



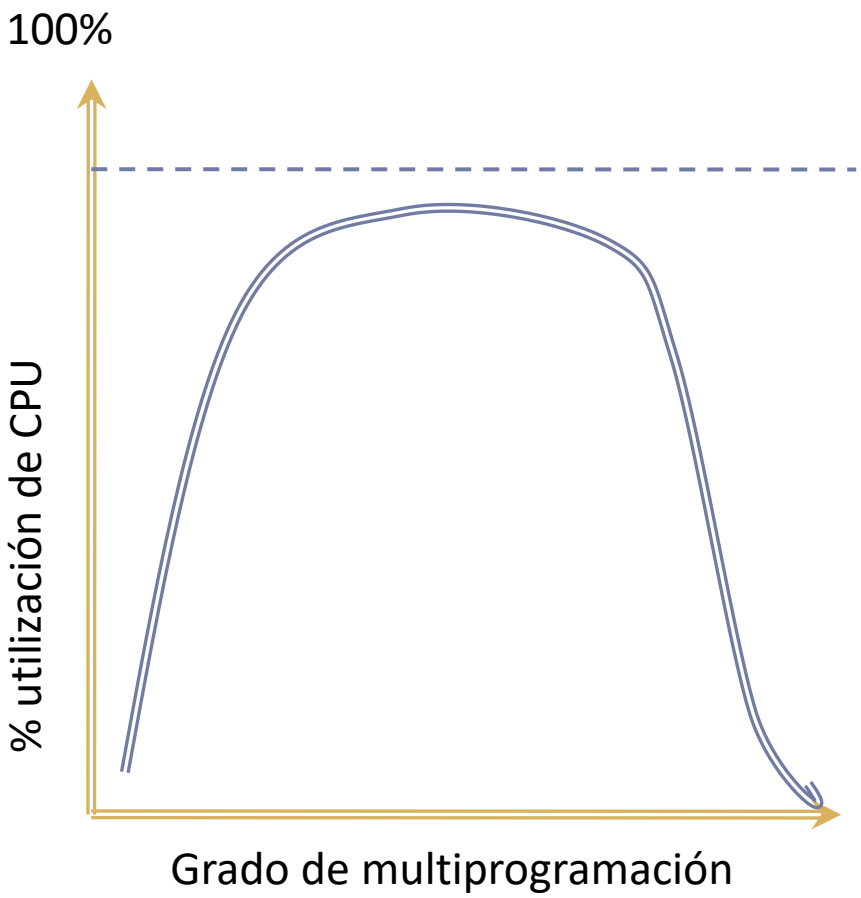
Rendimiento: Poca memoria física

- 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.



- Estado
- Lista/Cola
- Contexto

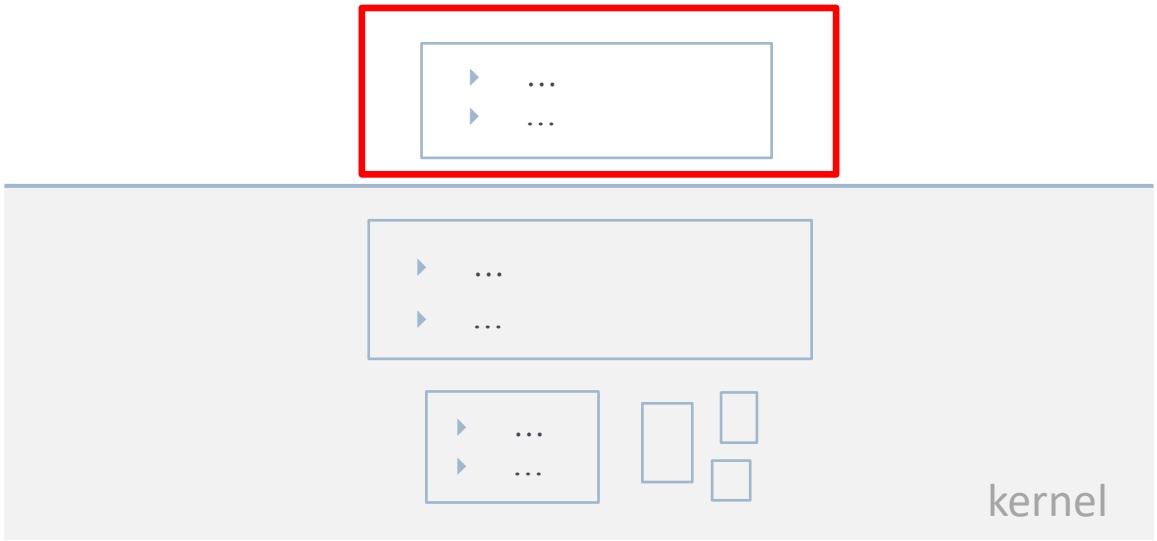
Ciclo de vida básico de un proceso

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

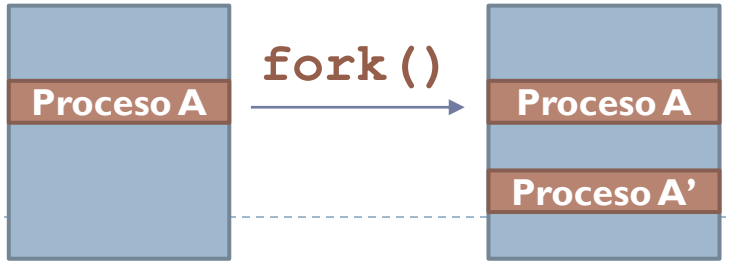
Contenidos

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.
   - Estructura de datos de procesos e hilos en el núcleo
   - Diseño e implementación de la multiprogramación y la multitarea en el núcleo

Servicios del sistema operativo
servicios POSIX de gestión de procesos

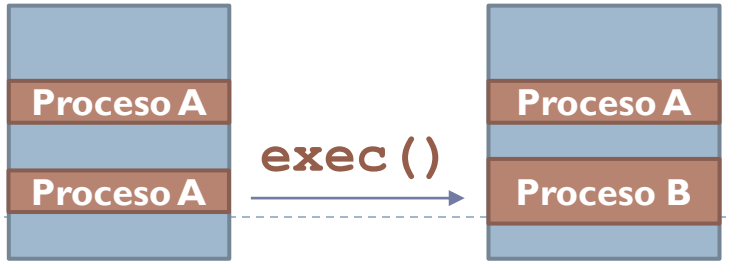


Servicio fork



Servicio	`#include <unistd.h>` **`pid_t fork(void);`**
Argumentos	
Devuelve	▫ -1 el caso de error. ▫ En el proceso padre: el identificador del proceso hijo. ▫ En el proceso hijo: 0
Descripción	▫ Duplica el proceso que invoca la llamada. ▫ Los procesos padre e 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.

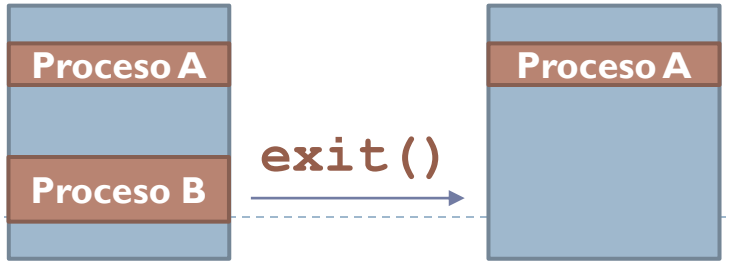
Servicio exec



Servicio	`#include <unistd.h>`
	`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[]);`

Argumentos	▫ path: Ruta al archivo ejecutable. ▫ file: Busca el archivo ejecutable en todos los directorios especificados por PATH
Devuelve	▫ Devuelve -1 en caso de error, **en caso contrario no retorna.**
Descripción	▫ Cambia la imagen del proceso actual. ▫ 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

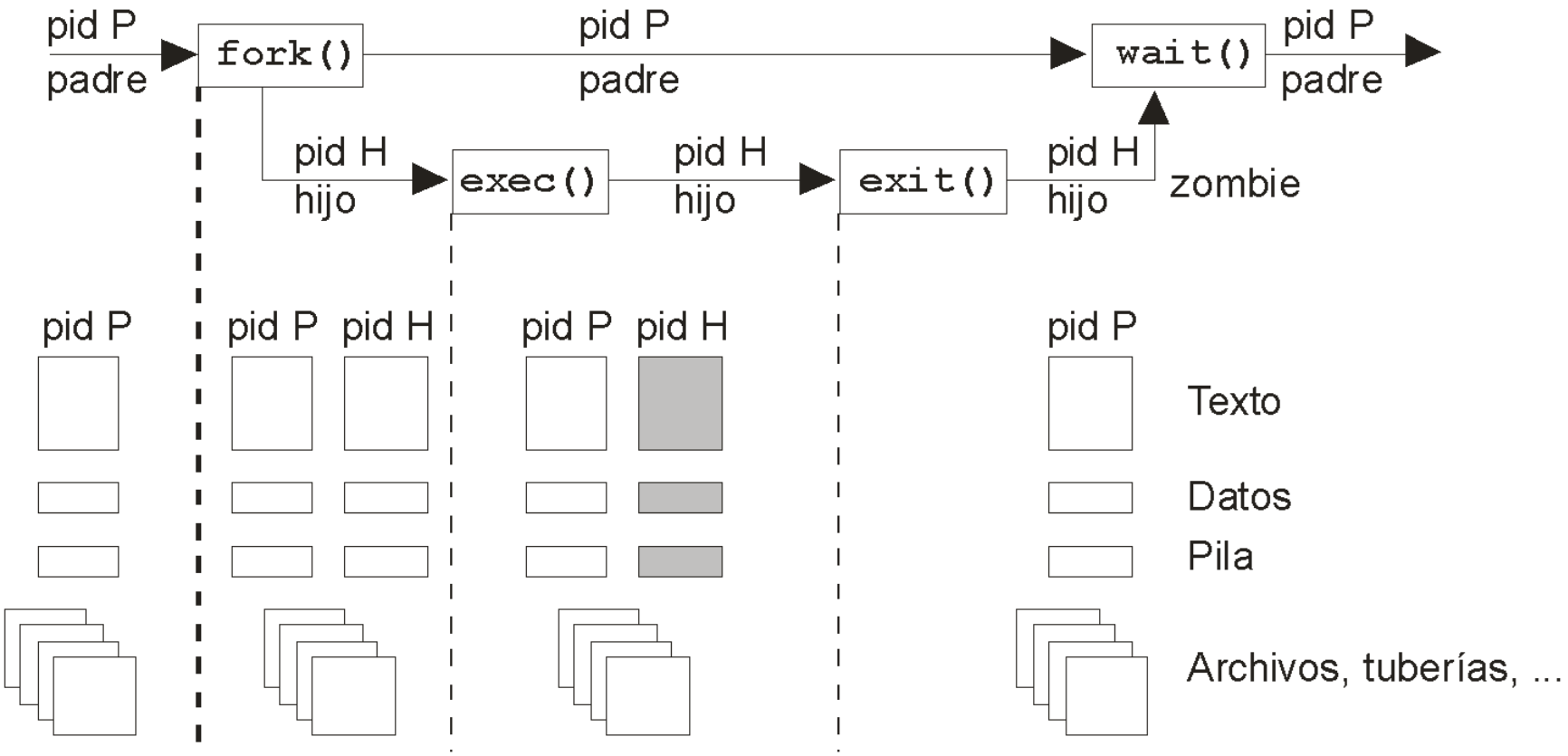
Servicio exit



Servicio	`#include <unistd.h>`
	
	`void exit(status);`

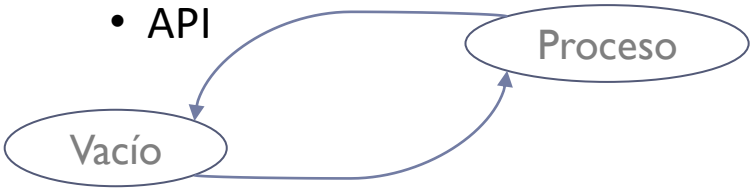
Argumentos	▫ status: valor que el padre recupera en la llamada wait()
Devuelve	
Descripción	▫ Finaliza la ejecución del proceso. ▫ Se cierran todos los descriptores de ficheros abiertos. ▫ Se liberan todos los recursos del proceso. ▫ Se libera el BCP del proceso.

Uso de los servicios fork, exec, wait y exit

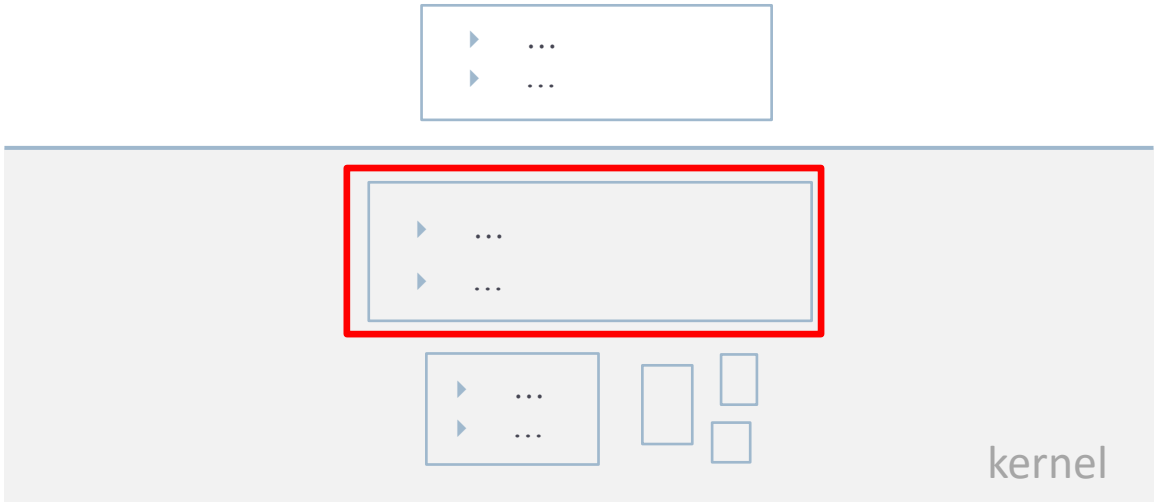


Servicios del sistema operativo
inicialización y finalización de procesos

Finalizar
- Razones
- API



Crear
- Razones
- API



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

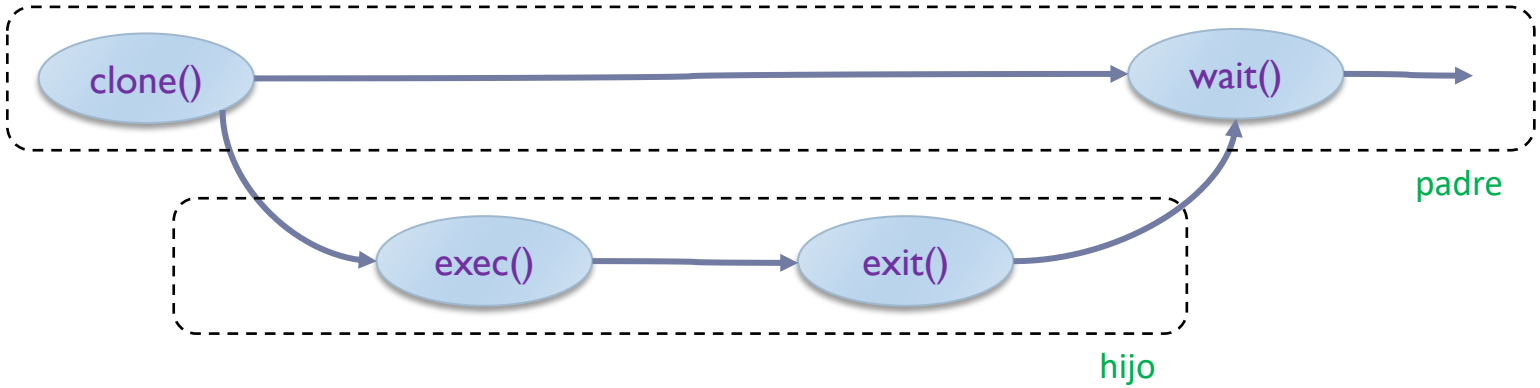
Finalización de procesos

- 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)

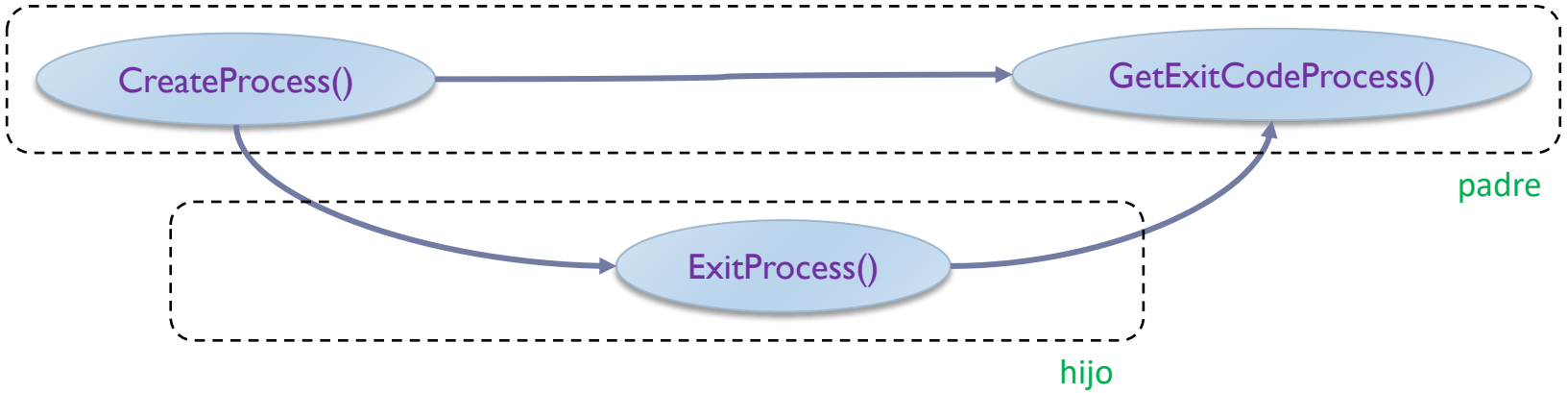
  - En Unix/Linux se usan señales como mecanismo
  - Se pueden capturar y tratar (salvo SIGKILL) para evitar finalizar involuntariamente

Creación y terminación de procesos
Llamadas al sistema

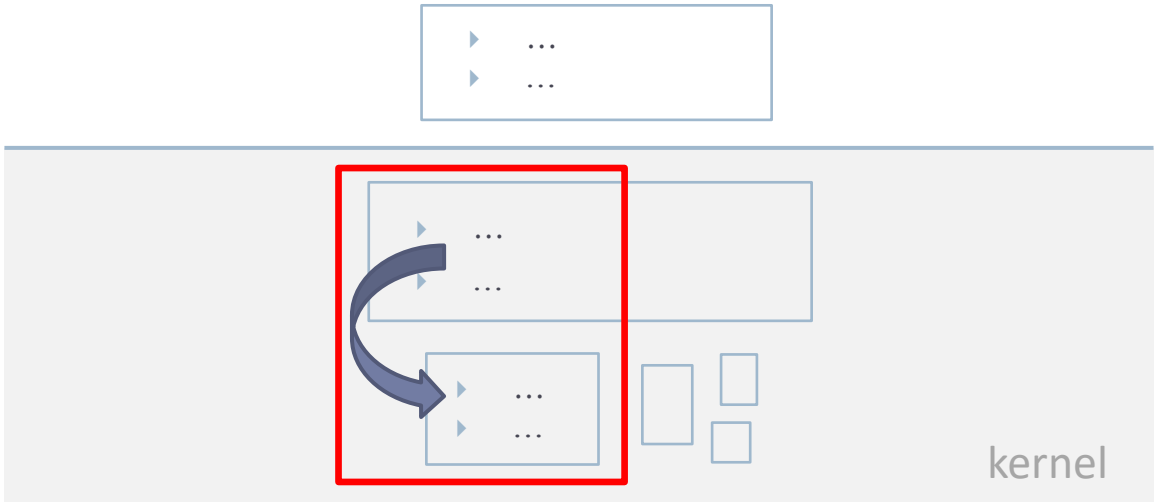
- Linux



- Windows



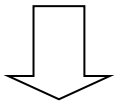
Servicios del sistema operativo
inicialización y finalización de procesos



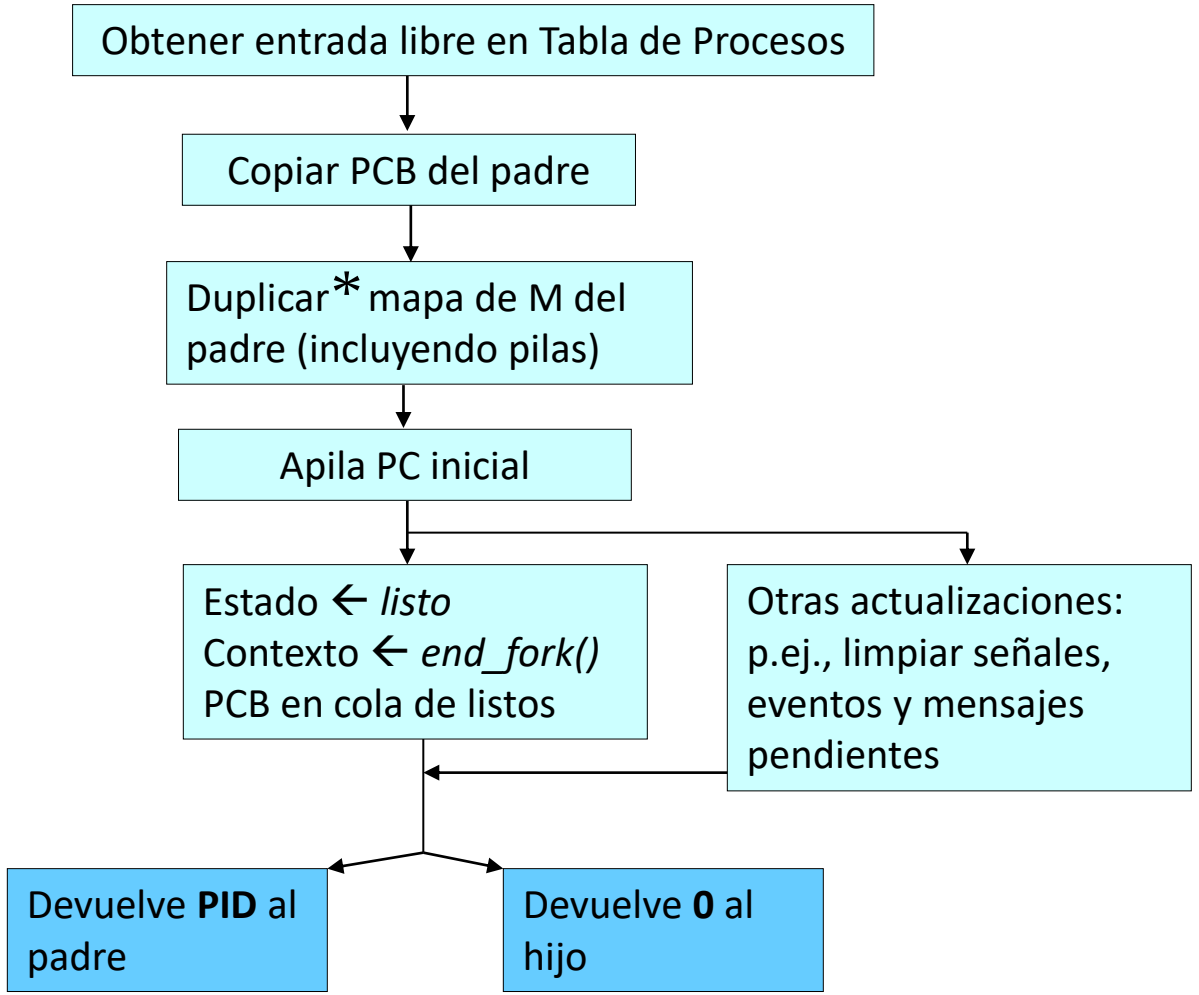
Creación de procesos

Linux: clone

clone:



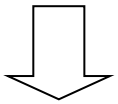
"Clona al proceso padre y da una nueva identidad al hijo"



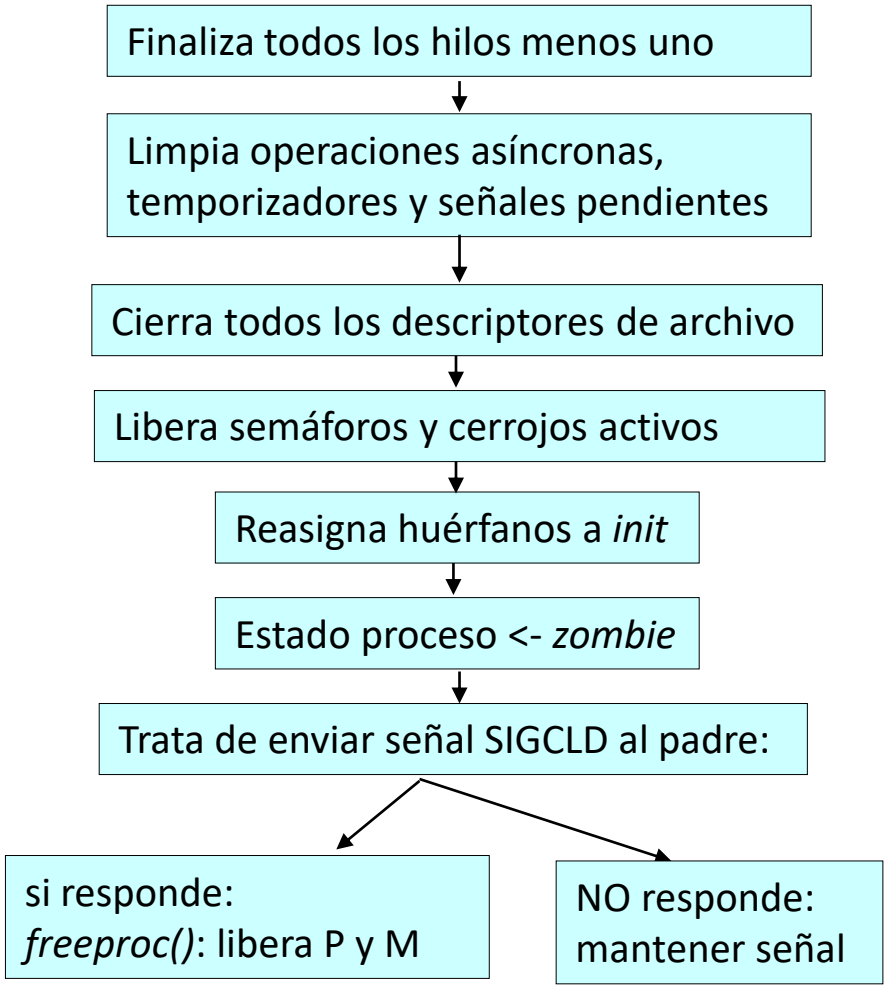
Terminación de procesos

Linux: exit

exit:



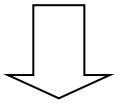
"Termina la ejecución de un proceso y libera los recursos"



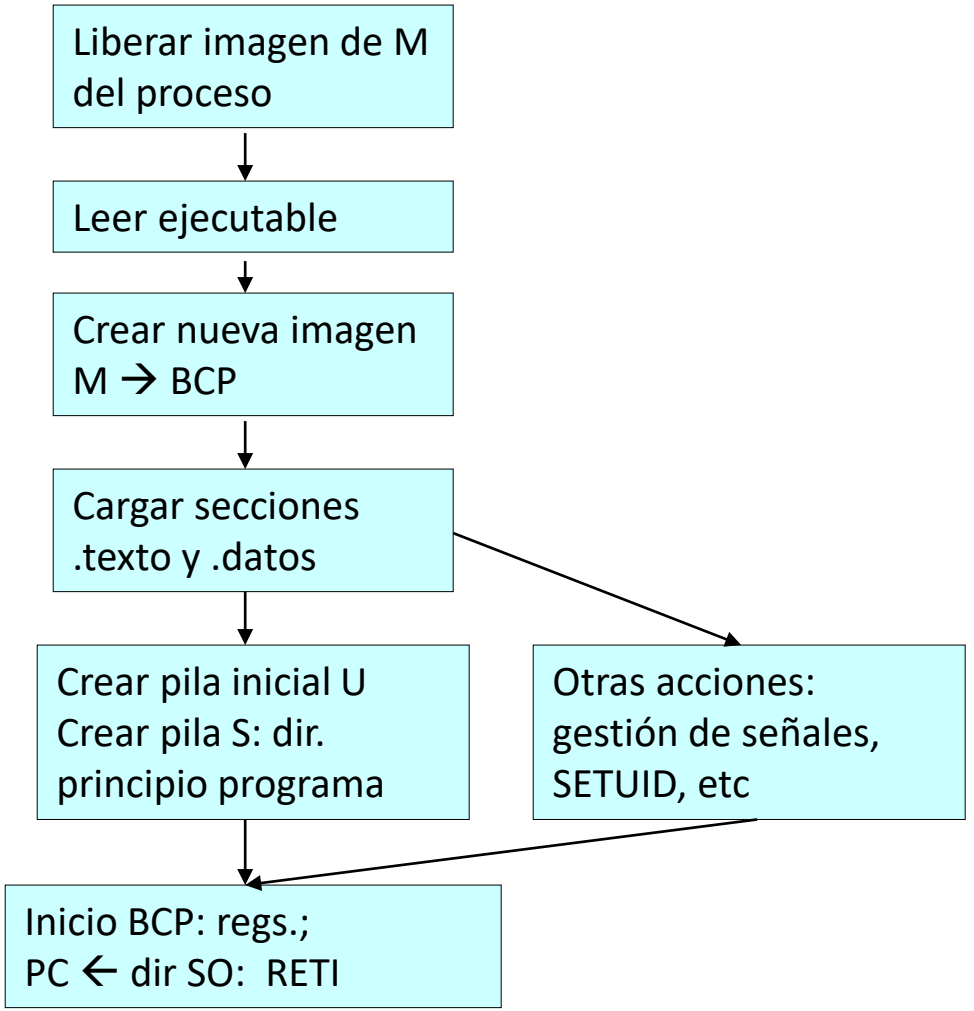
Cambio de imagen de un proceso

Linux: exec

exec:

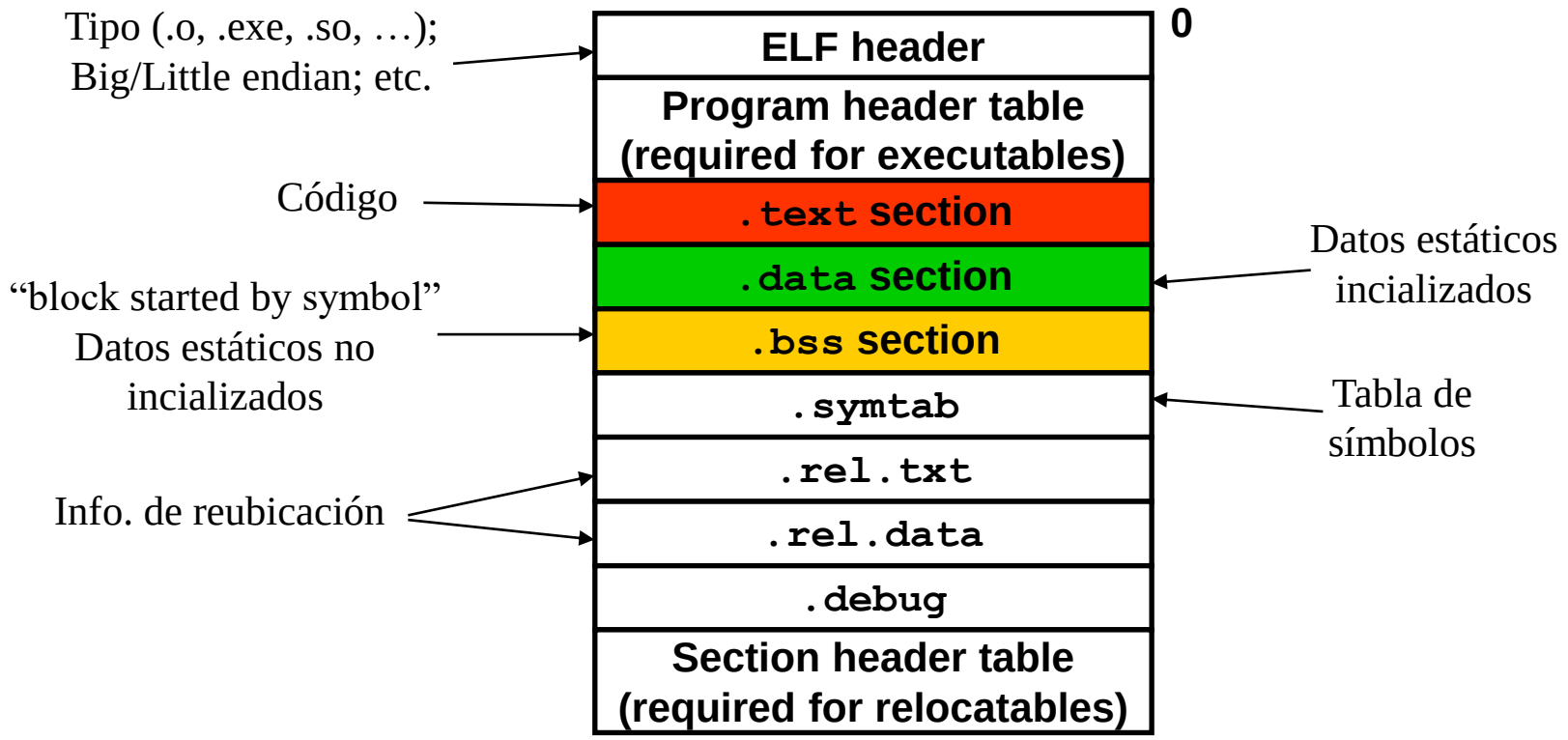


"Cambia la imagen de memoria de un proceso usando como 'recipiente' uno previo"

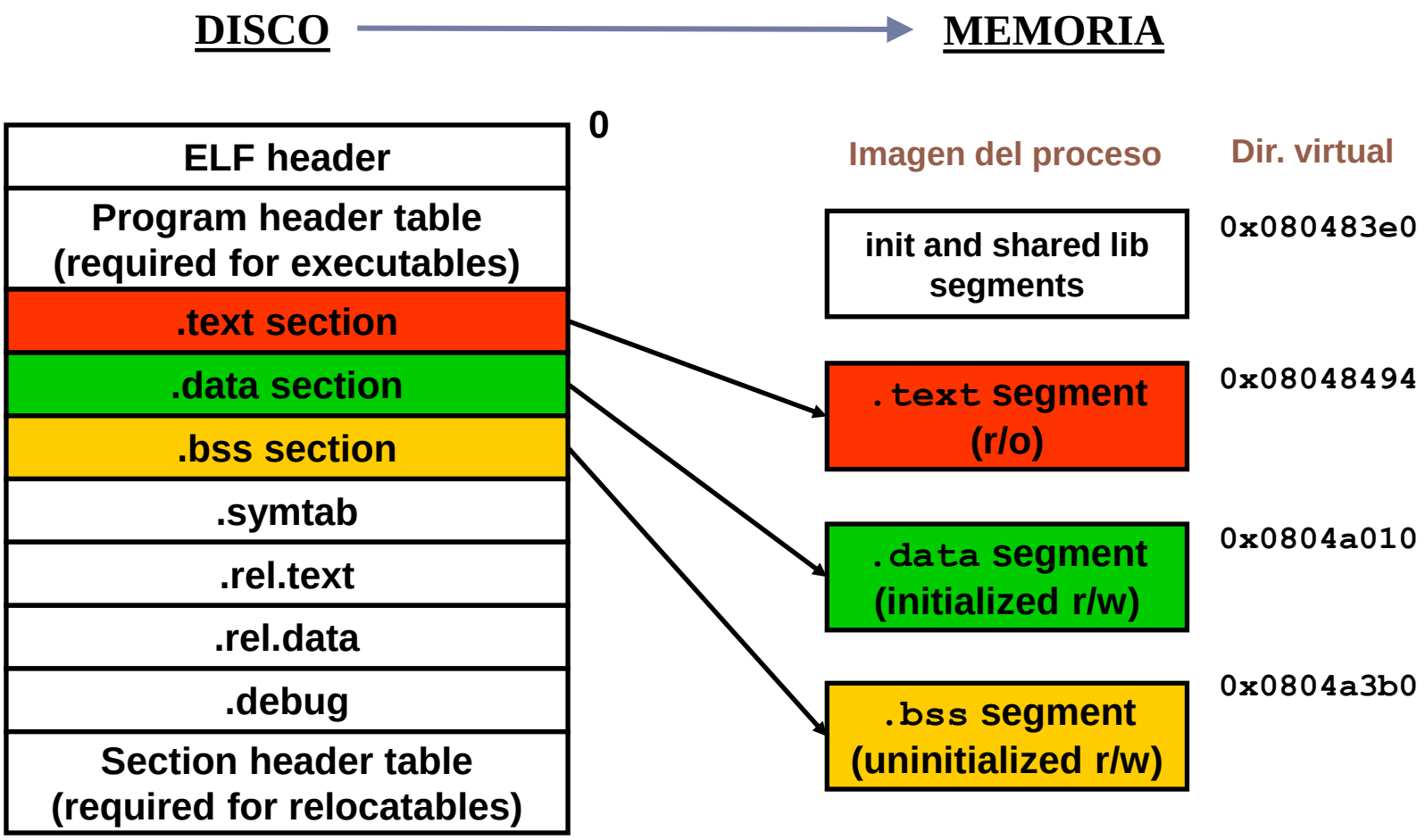


Cambio de imagen de un proceso
formato ELF de ejecutable en UNIX

- ELF: Executable and Linkable Format

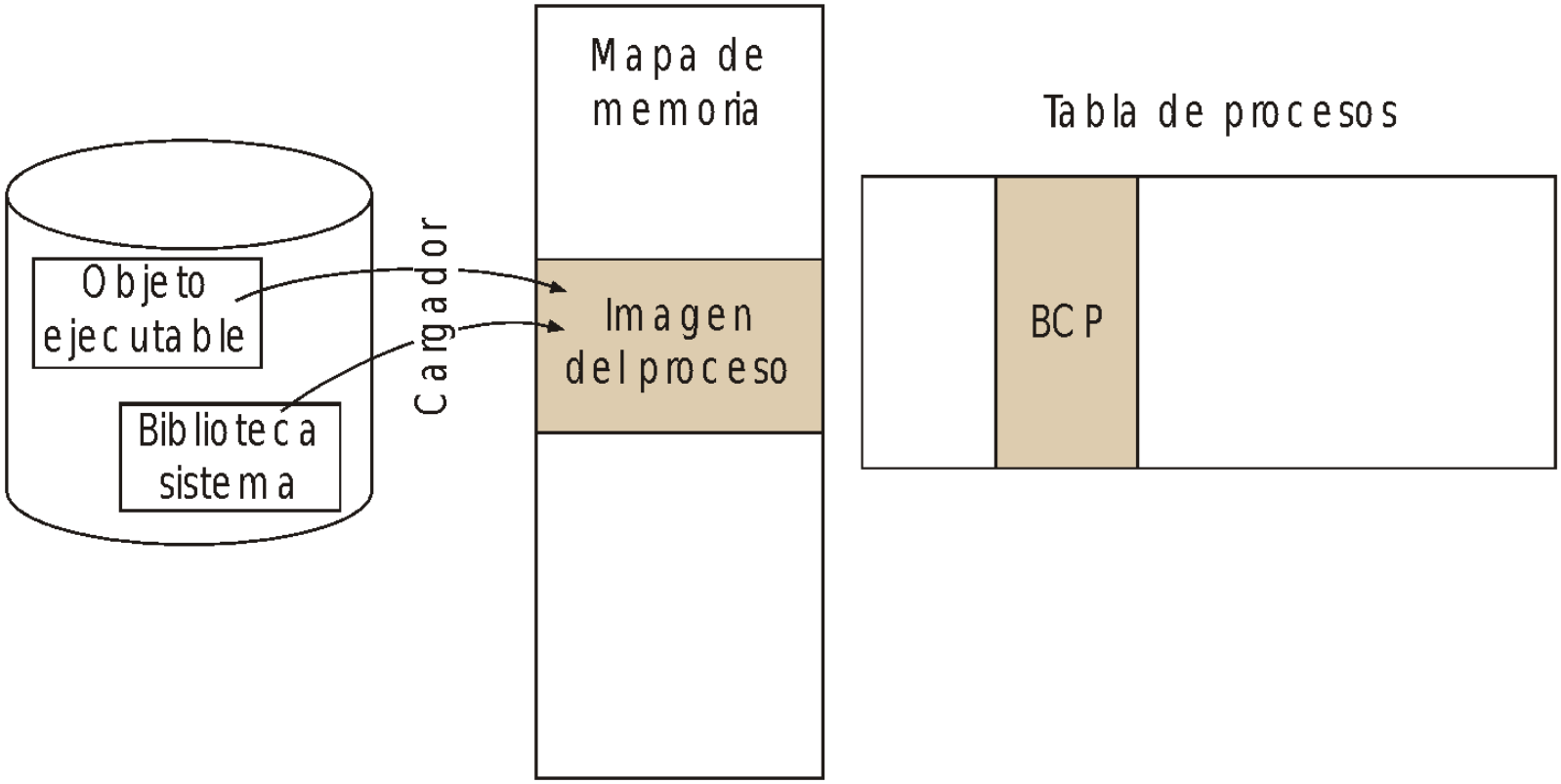


Cambio de imagen de un proceso
carga de ejecutable en memoria

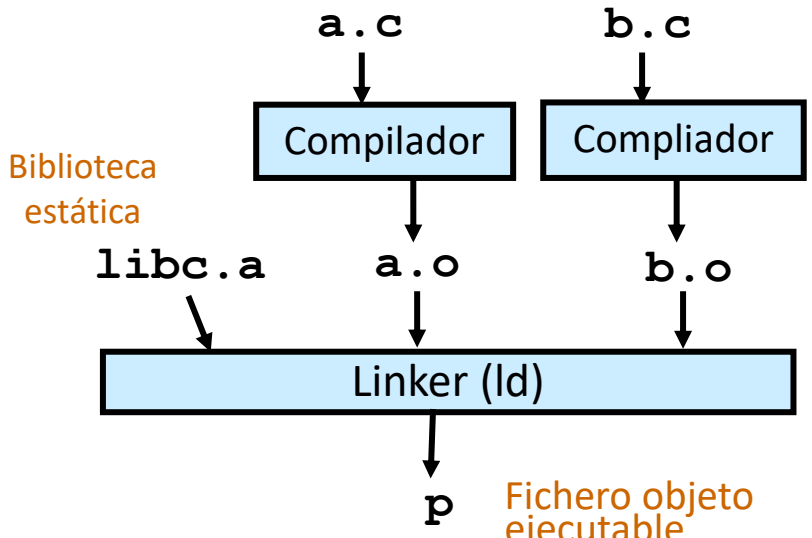


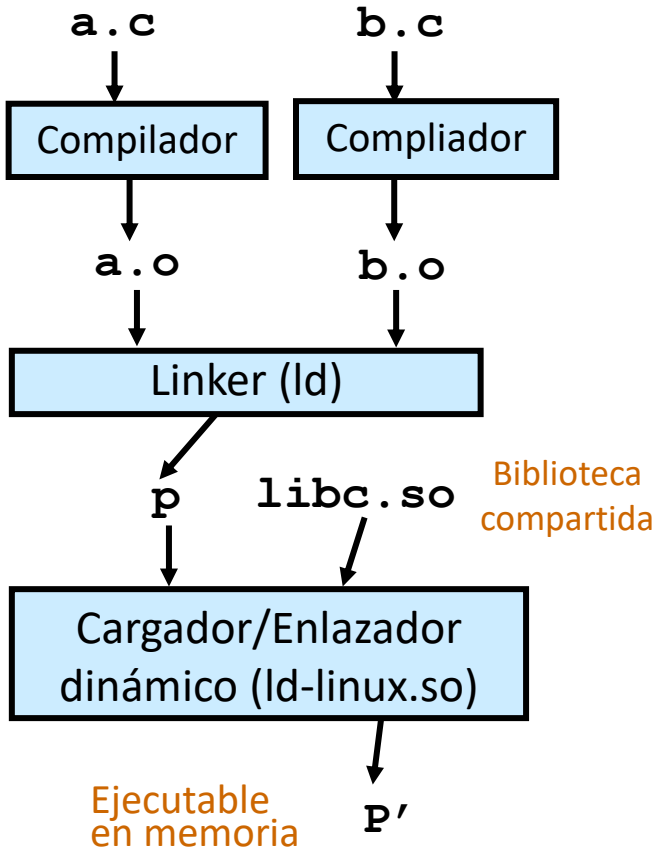
Cambio de imagen de un proceso

carga de ejecutable en memoria



Cambio de imagen de un proceso
generación de ejecutable





Bibliotecas estáticas

Bibliotecas dinámicas

Grupo ARCOS
Universidad Carlos III de Madrid

Lección 3
Procesos e hilos

Sistemas Operativos
Ingeniería Informática