Tema II

Descripción y control de procesos

UNED

Manuel Fernández Barcell

http://www.mfbarcell.es

Blog: http://prof.mfbarcell.es

2.2.1 CONCEPTO DE PROCESO

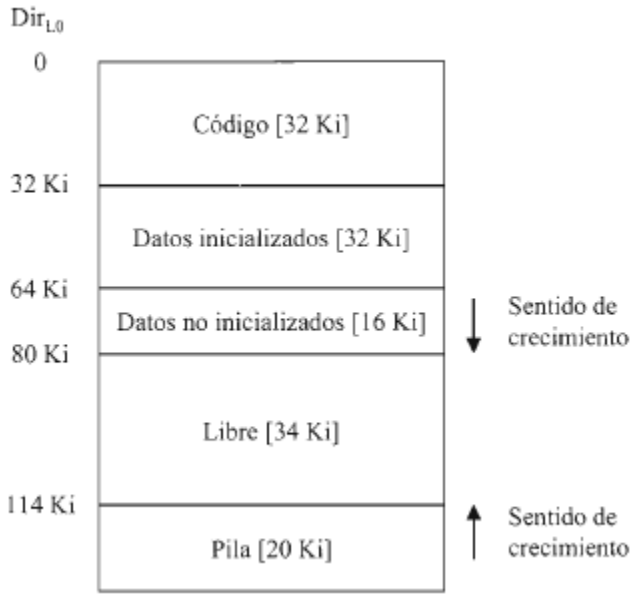
- Un programa es un archivo ejecutable que está en el almacenamiento secundario
  - Entidad pasiva o estática
- Es la unidad más pequeña de trabajo individualmente planificable por un sistema operativo
  - El proceso es un concepto dinámico que se refiere a un programa en ejecución, que sufre frecuentes cambios de estado y atributos
- Un proceso se ejecuta en primer plano (foreground)
  - Si los usuarios pueden interaccionar con el proceso durante su ejecución.
- Un proceso se ejecuta en segundo plano (background)
  - Si un usuario no puede interaccionar con el proceso durante su ejecución

Definición

- ¿Qué es un **proceso**?
  - **programa** en ejecución.
  - entorno de **protección**.
  - algo **dinámico**.
- Componentes básicos:
  - hebras/hilos de ejecución.
  - espacio de direcciones.
- La tarea fundamental de un SO es la gestión de procesos:
  - **creación**.
  - **planificación**.
  - **comunicación** (sincronización).
  - **finalización**.
- Un **programa** es...
  - una lista de **instrucciones** (especie de receta de cocina).
  - algo **estático**.
  - **varios** procesos pueden ejecutar un mismo programa.
- Puede **lanzar**, o ser lanzado por, otros procesos.

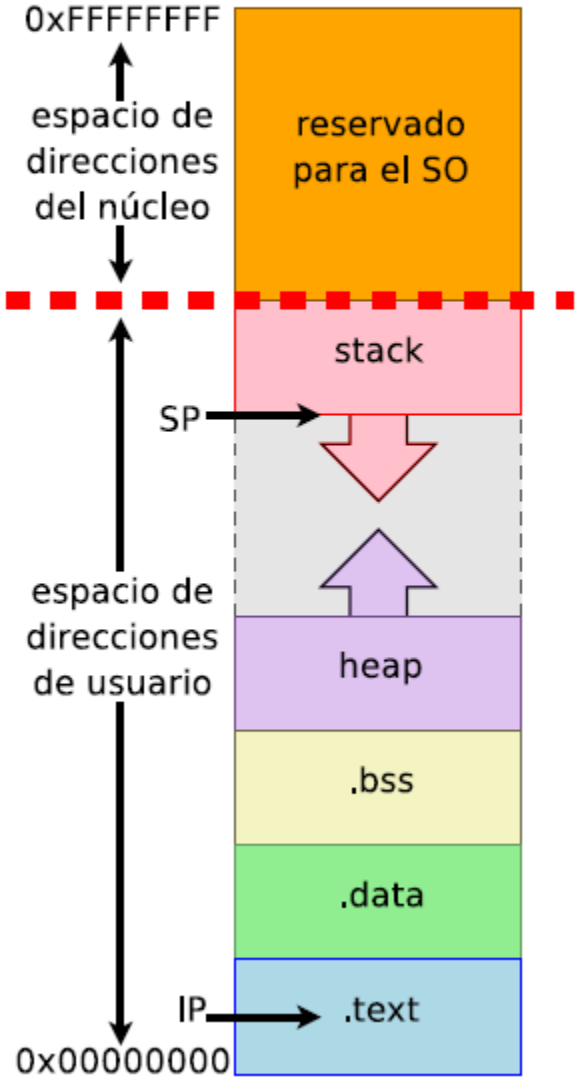
ESPACIO DE DIRECCIONES DE MEMORIA LÓGICA O VIRTUAL DE UN PROCESO

- Imagen del proceso o espacio de direcciones de usuario se divide en regiones
  - La región de código (corresponde a zona de código del archivo ejecutable)
  - La región de datos (corresponde a zona de datos del archivo ejecutable)
    - La región de datos inicializados (tamaño fijo)
    - La región de datos no inicializados (tamaño variable)
  - (Espacio libre)
  - La región de pila
    - Controlada por el puntero de pila
    - Tamaño variable en ejecución
      - La pila crece de la direcciones alta a las más baja (sentido contrario



Ejemplo: Espacio de direcciones en Linux

- Espacio de direcciones lógicas a las que puede acceder un proceso:
  - código: .text
  - datos:
    - inicializados: .data
    - sin inicializar: .bss
    - dinámicos: heap
  - pila: stack
- En una parte del mismo espacio de direcciones del proceso se ejecuta el núcleo de Linux.
- Divisiones típicas: 2GB/2GB y 3GB/1GB.



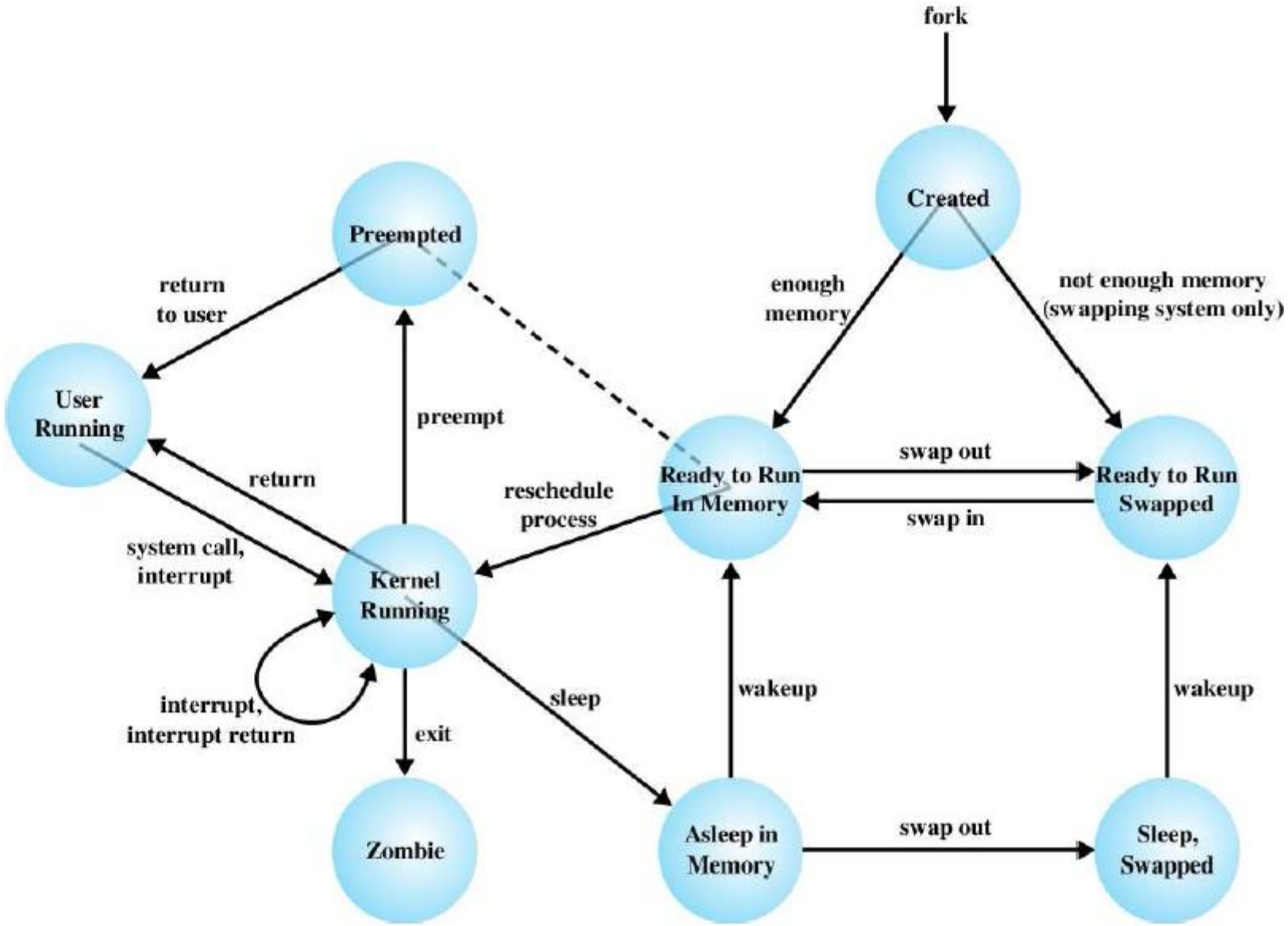
TIPOS DE PROCESOS

- Procesos de usuarios
  - Invocados por los usuarios
  - Se ejecutan en modo usuario
    - Excepto cuando realizan llamadas al sistema
  - En primer o segundo plano
- Procesos demonios
  - Tareas periódicas iniciados por el S.O
  - Se ejecutan en modos usuario
- Procesos del sistema Operativo
  - Se ejecutan en modo supervisor y normalmente en segundo plano

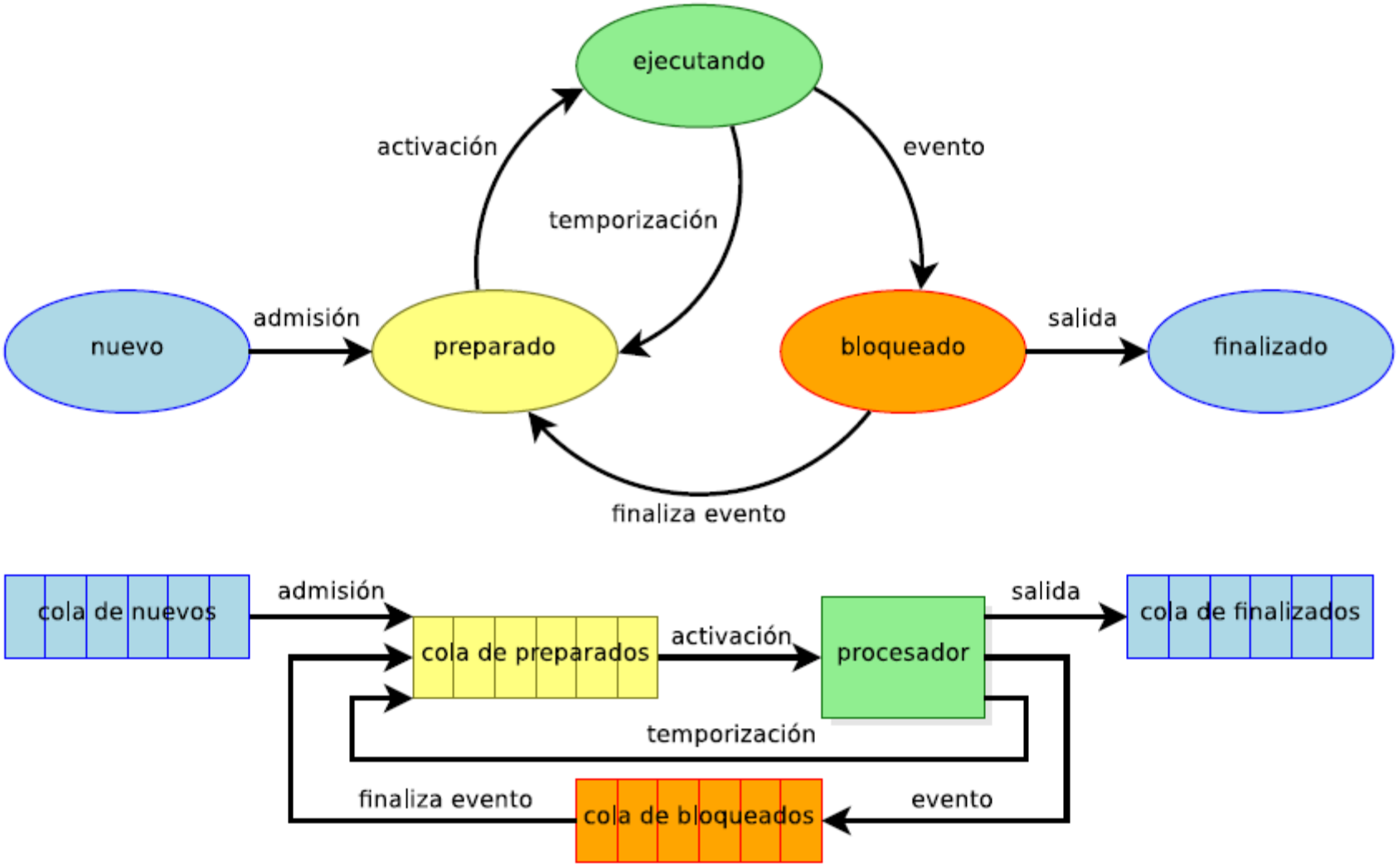
ESTADO DE LOS PROCESOS

- El tiempo de vida de un proceso puede ser conceptualmente dividido en un conjunto de estados que describen el comportamiento
- NUEVO:
  - Existe, pero faltan estructuras por crear
- Preparado:
  - Listo para ejecutarse, espera a que un/el procesador quede libre y sea planificado
- EJECUTÁNDOSE:
  - Se está ejecutando en ese instante
- Bloqueado o suspendido:
  - Espera a que se cumpla alguna condición (termine alguna operación de E/S...)
- TERMINADO:
  - Termina su ejecución o bien el s.o. ha detectado un error fatal

Diagrama de transiciones entre estados en UNIX

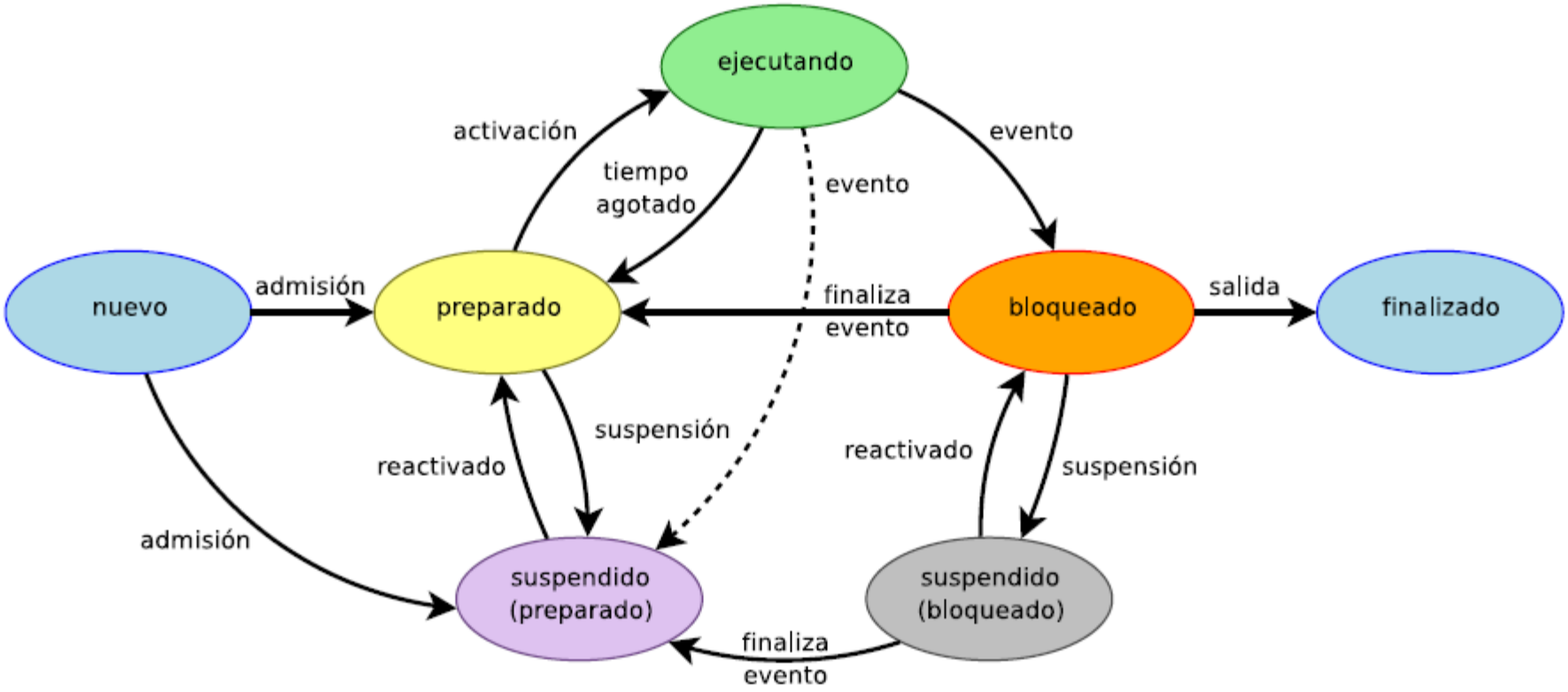


Modelo de 5 estados



Modelo de 6 estados

Modelo de 7 estados



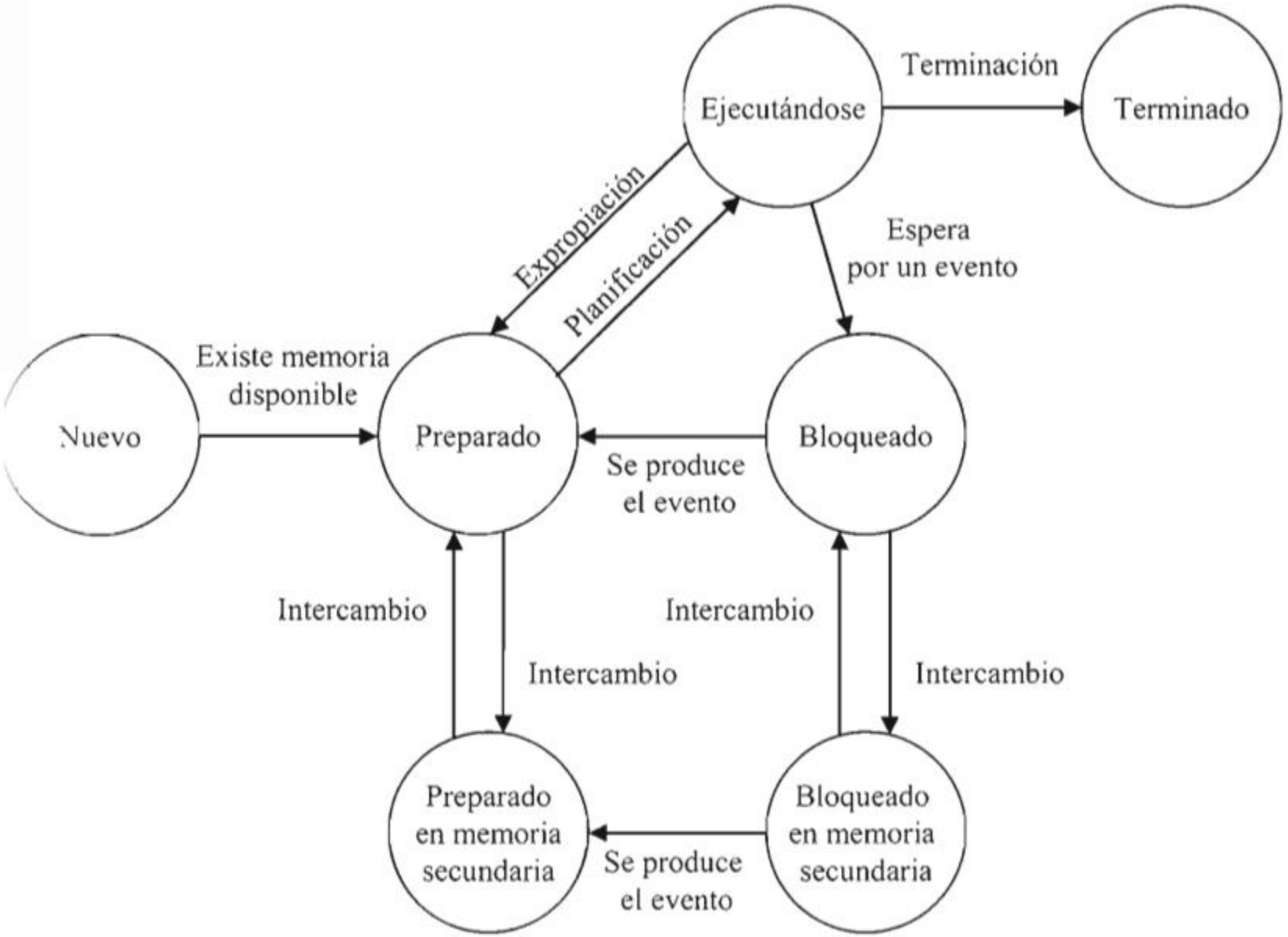


Figura 2.3 – Diagrama de transición de estados de un sistema operativo con siete posibles estados por proceso

CONTROL DE PROCESOS

Estructuras de control del sistema operativo

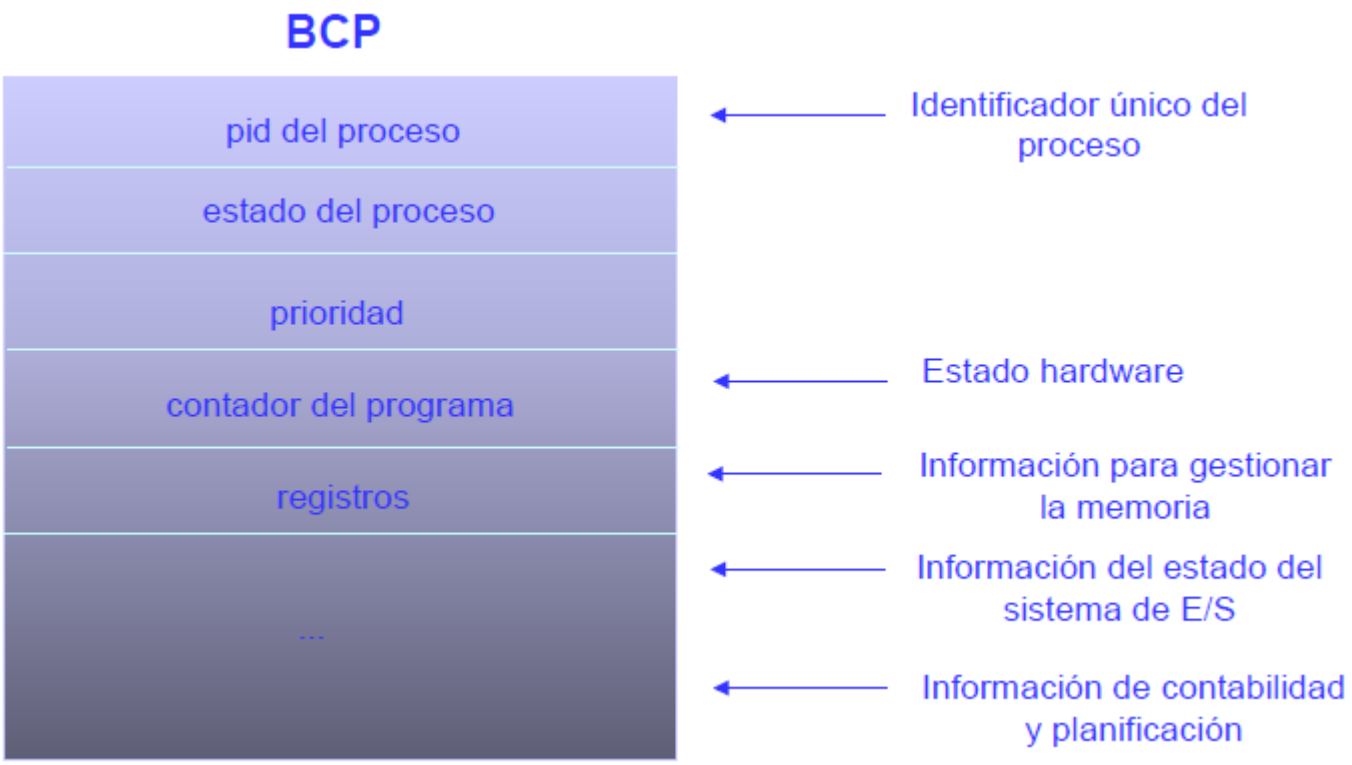
- Para **gestionar** procesos y recursos el SO debe disponer de **información** sobre estos.
- El SO mantiene **tablas** sobre cada entidad que gestiona:
  - Tablas de **memoria**: principal y secundaria, protección, tradución.
  - Tablas de **E/S**: dispositivos y canales, estado de las operaciones.
  - Tablas de **ficheros**: existencia, atributos, localización,...
  - Tablas de **procesos**: localización y atributos.
- Las tablas anteriores no suelen estar separadas sino entrelazadas.
- Normalmente las tablas se inicializan al arrancar el sistema mediante autoconfiguración.
- Ejemplo: Linux, `struct task_struct` en `sched.h`.

Estructuras de control de procesos

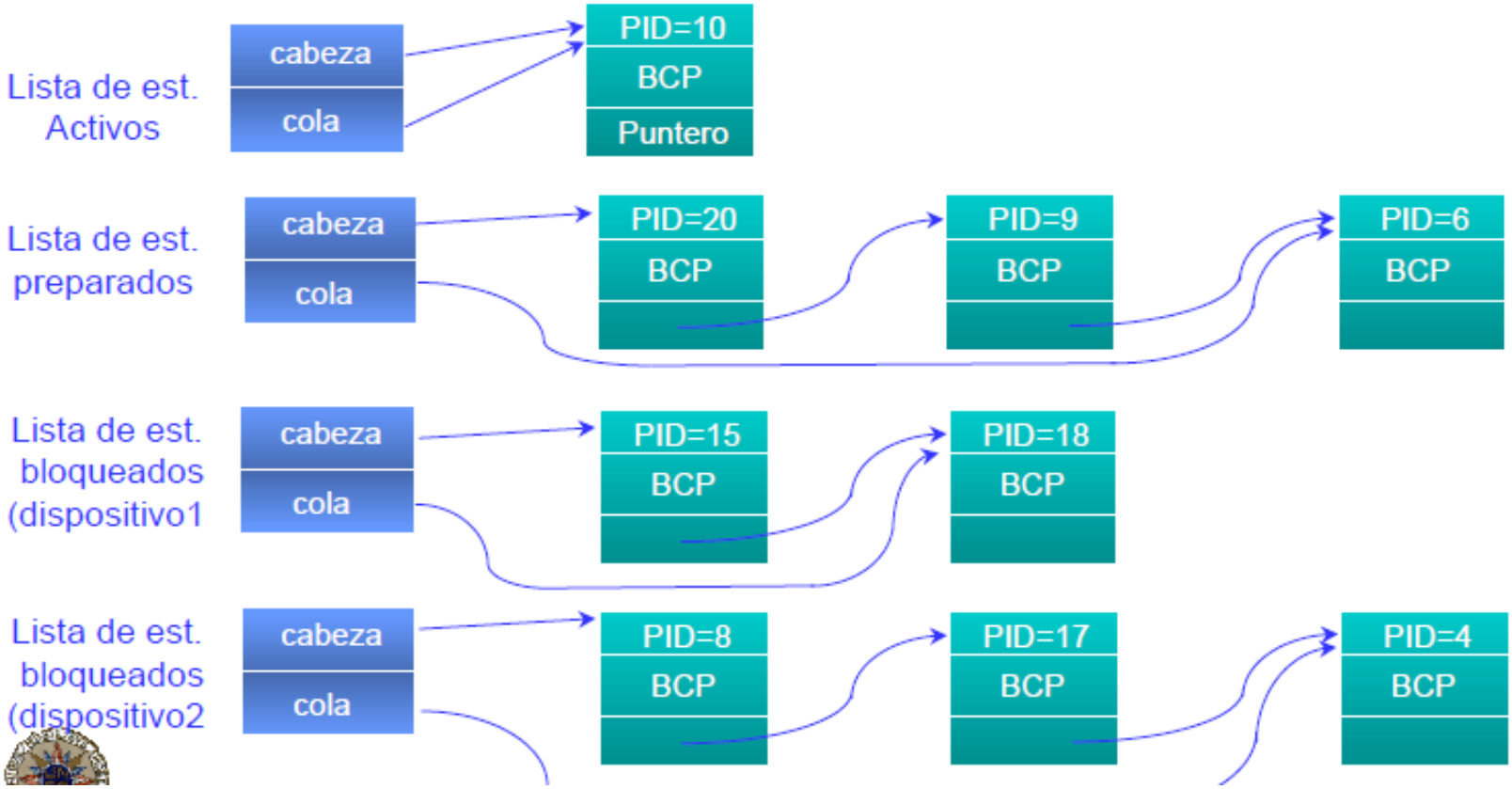
- Representación física de un proceso:
  - **Imagen del proceso**:
    - programa a ejecutar.
    - espacio de direcciones disponible para código, datos y pila.
  - **Bloque de Control del Proceso (PCB)** o **descriptor de proceso**:
    - atributos para la gestión del proceso por parte del SO.
    - **estructura de datos más importante del SO.**
- Atributos de un proceso:
  - **Identificación** del proceso: identificadores del proceso, proceso padre, usuario.
  - Estado del **procesador**: registros de propósito general, de estado y control, puntero de pila.
  - Información de **control** del proceso: estado, planificación, estructuración, comunicación y sincronización, privilegios, gestión de memoria, control de recursos y utilización.

Bloque de control de procesos PCB

- La tabla de procesos es una estructura del sistema operativo que almacena información de control relevante sobre cada proceso existente.
  - Cada entrada de la tabla de procesos se denomina bloque de control del proceso (*process control block*, PCB)
- Campos del BCP
  - Identificador de procesos
  - Identificador de usuario
  - Estado del proceso
  - Contenido de algunos registros del procesador
  - Información del planificador de procesos
  - Información de localización de la memoria principal asignada al proceso
  - Información de tipo estadístico
  - Información de estado de E/S
  - Información sobre eventos por la que el proceso ha entrado en estado bloqueado


BCP
pid del proceso
estado del proceso
prioridad
contador del programa
registros
...
Identificador único del proceso
Estado hardware
Información para gestionar la memoria
Información del estado del sistema de E/S
Información de contabilidad y planificación

- El s.o. mantiene listas del sistema:
  - Una lista para los procesos que están en estado preparado
  - Una lista de los procesos en estado bloqueado
    - El proceso puede estar en una única lista de estados bloqueados o en una lista de estados suspendidos ligada en exclusiva a un dispositivo o evento
  - Una lista de los procesos en estado activo o de en ejecución
    - Si el sistema es monoprocesador, sólo posee una entrada
- El planificador es el que gestiona el paso de los procesos de una lista a otra


Lista de est. Activos
cabeza
cola
PID=10
BCP
Puntero
Lista de est. preparados
cabeza
cola
PID=20
BCP
PID=9
BCP
PID=6
BCP
Lista de est. bloqueados (dispositivo1
cabeza
cola
PID=15
BCP
PID=18
BCP
Lista de est. bloqueados (dispositivo2
cabeza
cola
PID=8
BCP
PID=17
BCP
PID=4
BCP

Creación de procesos

Pasos en la creación de un proceso:

1. Asignar un **identificador de proceso único**.
2. **Reservar espacio** para el proceso:
   - estructuras de datos del SO (**PCB**).
   - imagen del proceso.
3. **Inicialización** del bloque del control del proceso (**PCB**).
   - ppid, estado, ip, sp, prioridad, E/S,...
4. Establecimiento de **enlaces** adecuados:
   - cola de trabajos.
5. Creación o expansión de otras estructuras de datos:
   - auditoría, monitorización, análisis de rendimiento,...

Creación de procesos

- Salvo sistemas extremadamente simples los SO deben tener mecanismos para la creación de nuevos procesos.
- Posibles causas de la creación de un procesos:
  1. Inicialización del sistema.
     - interactivos / no interactivos.
     - primer / segundo plano.
  2. Llamada al sistema para crear un proceso.
     - `fork() + exec() / CreateProcess()`.
  3. Petición de usuario.
     - lanzamiento de una nueva aplicación desde el interfaz de usuario.
  4. Inicio de un proceso por lotes.
     - sistemas de colas de trabajos en servidores.

Finalización de procesos

- Una vez creados los procesos se ejecutan y, generalmente, realizan la tarea para la que se lanzarón.
- Causas de finalización de un proceso:
  - Voluntarias:
    - Terminación normal: la mayoría de los procesos realizan su trabajo y devuelven el control al SO mediante la llamada al sistema `exit()` / `ExitProcess()`.
    - Terminación por error: falta argumento,...
  - Involuntarias:
    - Error fatal: instrucción privilegiada, excepción de coma flotante, violación de segmento,...
    - Terminado por otro proceso: mediante la llamada al sistema `kill()` / `TerminateProcess()`

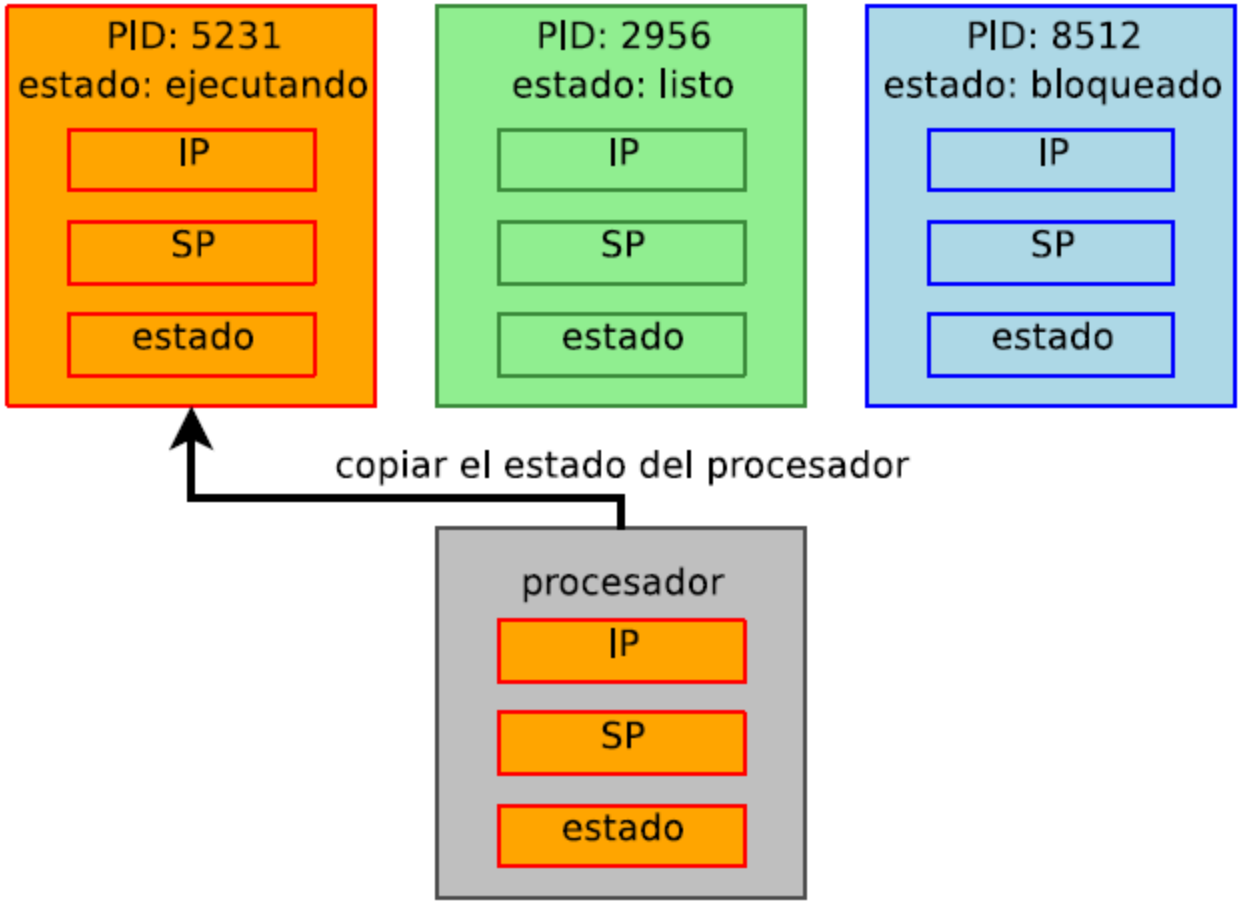
Cambio de proceso

- Es una de las operaciones más frecuentes:
  - Interrumpir la ejecución de un proceso para que pase a ejecutarse otro proceso
    - Cambio de proceso o cambio de contexto
- Causas
  - El proceso pasa al estado de bloqueado
  - Terminación del proceso
  - Atención a interrupción
  - Tiempo de ejecución excedida
- Tiempo de conmutación
  - Tiempo empleado en la operación de cambio de proceso

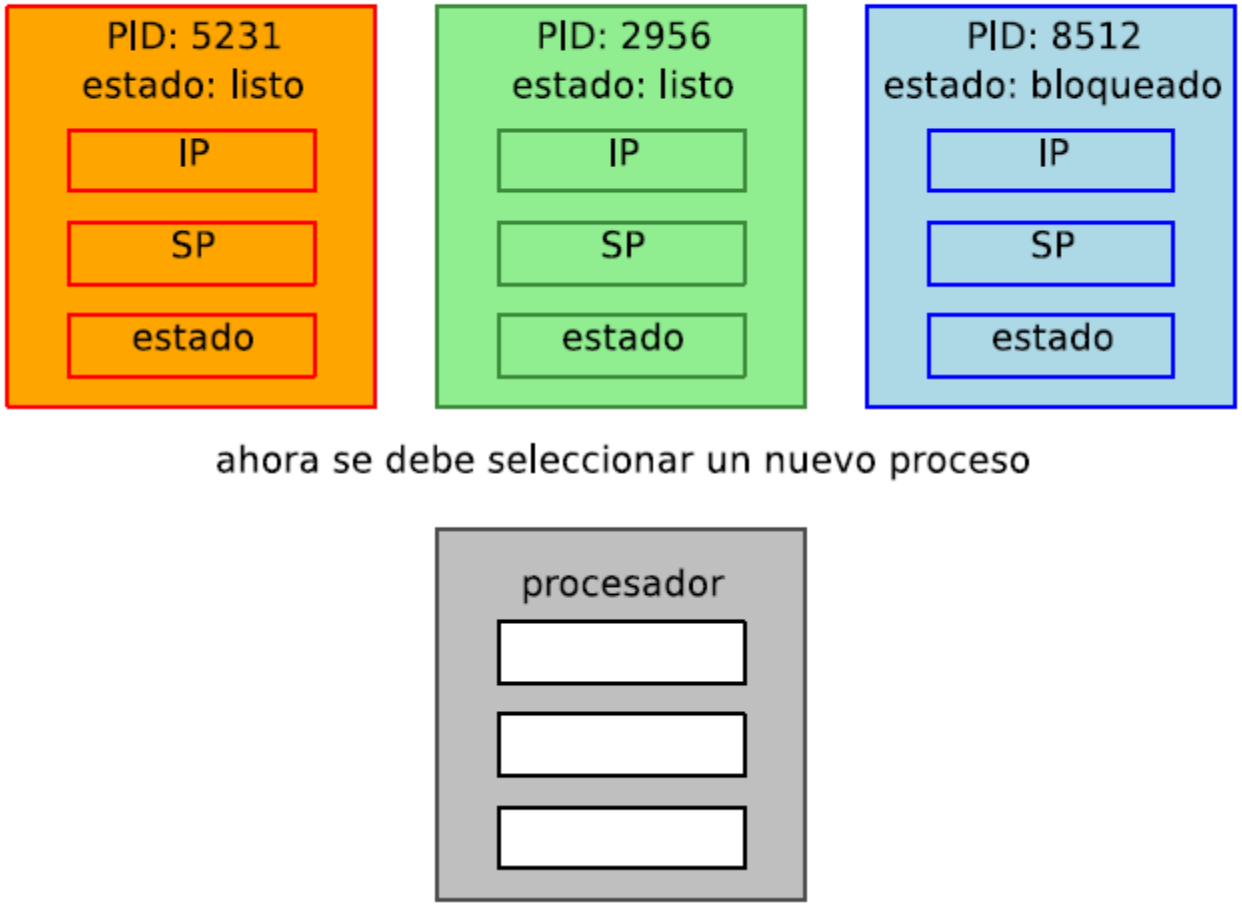
Pasos para el cambio de proceso

1. Salvar el estado del procesador.
2. Actualizar el estado del bloque de control del proceso.
   - Como mínimo cambiar el estado (ej: ejecutando → preparado).
3. Mover el PCB a la cola adecuada (ej: preparado).
4. Seleccionar el nuevo proceso a ejecutar.
5. Actualizar el estado del bloque de control del proceso.
   - Como mínimo cambiar el estado (ej: preparado → ejecutando).
6. Actualizar las estructuras de datos de gestión de memoria.
7. Restaurar el estado del proceso al que tenía en el momento de abandonar el estado ejecutando.

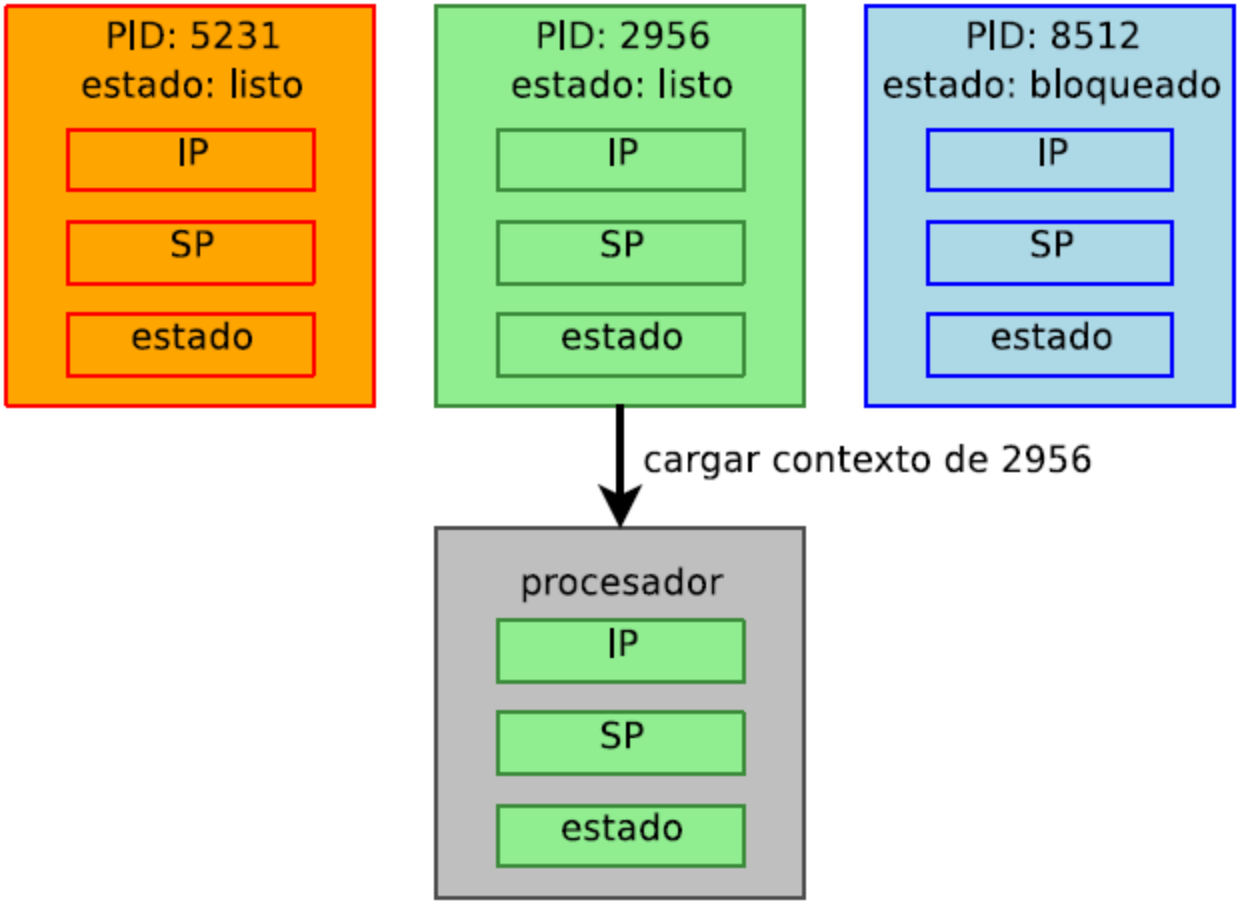
Cambio de proceso



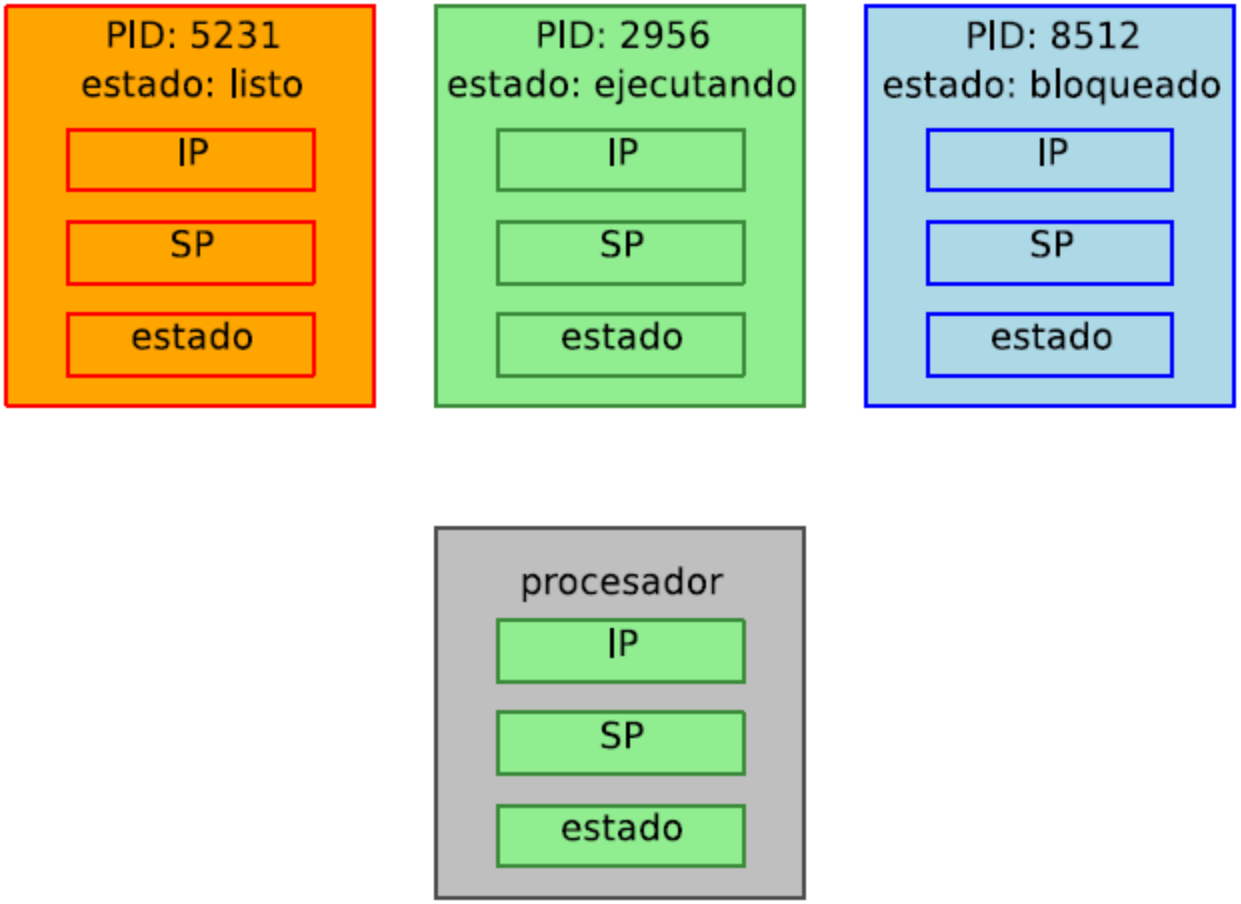
Cambio de proceso



Cambio de proceso

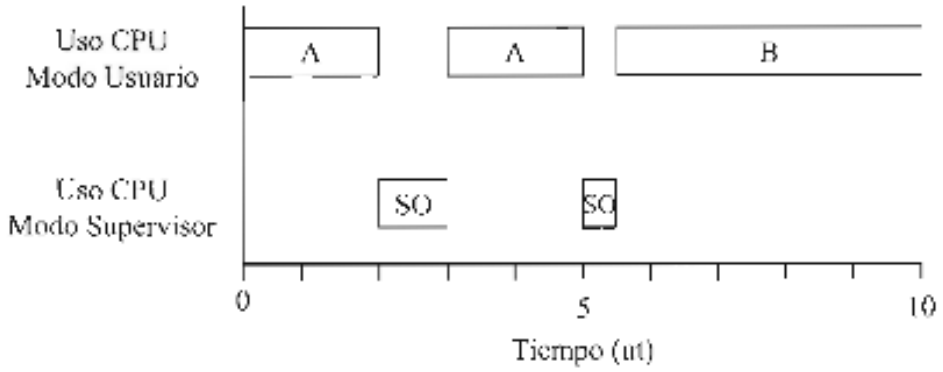


Cambio de proceso



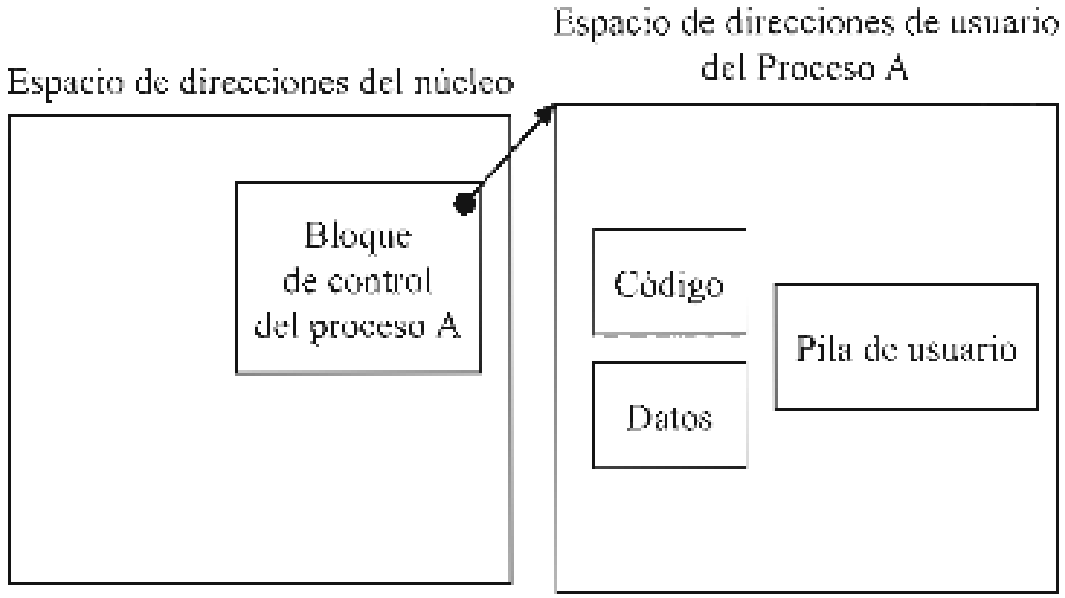
2.4 Ejecución del sistema operativo

- En un computador con un único procesador, el uso del mismo se va alternando entre
  - El sistema operativo, (se ejecuta en modo supervisor)
  - Los distintos tipos de procesos que se ejecutan en modo usuario.
- Sobrecarga del sistema (*overhead*)
  - El tiempo que el procesador se encuentra ocupado ejecutando código del sistema operativo asociados a tareas y servicios de administración que no se puede contabilizar a ningún proceso en particular

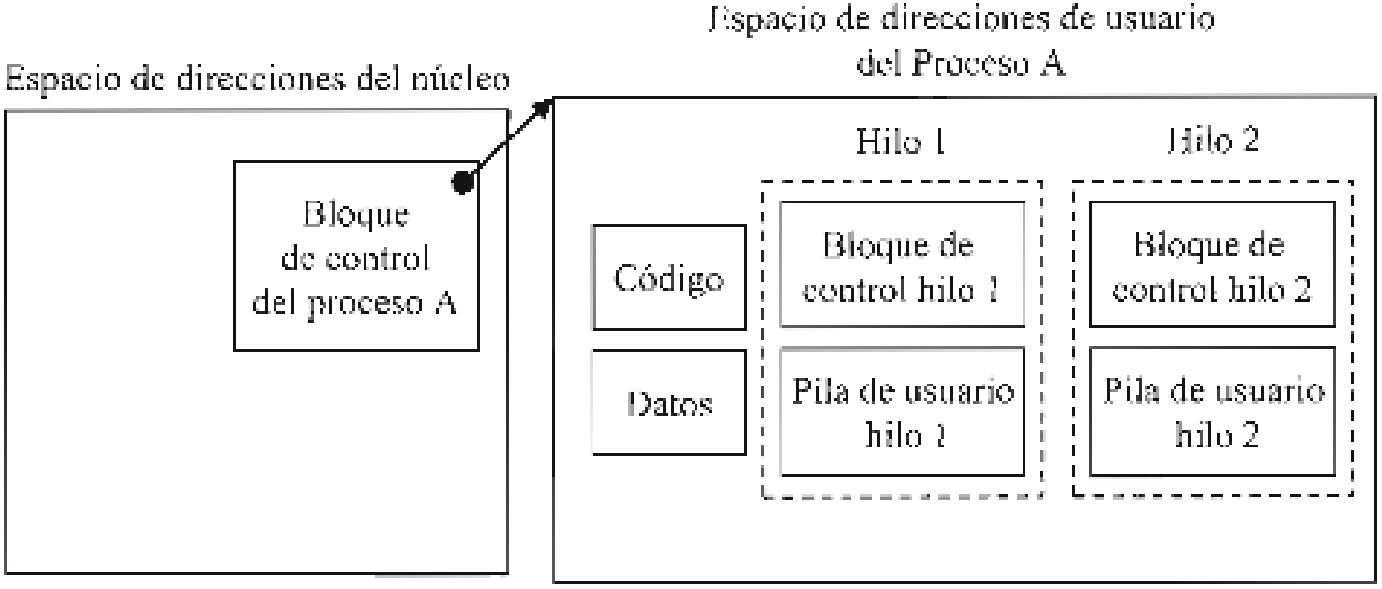


Procesos multihilos

- Un proceso se puede considerar como una entidad computacional básica que tiene asignado un conjunto de recursos y que durante su existencia sigue una determinada ruta o traza de ejecución.
- Un proceso queda caracterizado por dos elementos
  - Conjunto de recursos asignados
    - Espacio de direcciones (código, datos y pila), archivos abiertos, información de contabilidad…
  - Traza de ejecución (hilo o hebra)
    - Las instrucciones que ejecuta el proceso
    - El hilo tiene asignado, un contador de programa, registros, pila
    - Es la unidad elemental de asignación del procesador


Espacio de direcciones del núcleo
Espacio de direcciones de usuario del Proceso A
Bloque de control del proceso A
Código
Datos
Pila de usuario

(a)


Espacio de direcciones del núcleo
Espacio de direcciones de usuario del Proceso A
Bloque de control del proceso A
Hilo 1
Hilo 2
Código
Datos
Bloque de control hilo 1
Pila de usuario hilo 1
Bloque de control hilo 2
Pila de usuario hilo 2

(b)

Los sistemas operativos multihilo

- Utilizan los procesos como unidades de gestión de recursos
- Utilizan a los hilos como unidad de asignación del procesador .
  - Un hilo se puede definir como una unidad elemental de asignación del procesador que sigue una determinada traza de ejecución y que tiene asignado
  - Una pila de usuario,
  - Un espacio de almacenamiento y
  - Un bloque de control de hilo.

- Proceso en un SO "*tradicional*" = espacio de direcciones + hebra de control.
- Proceso = contenedor de recursos + hebra de ejecución.
- Una hebra se ejecuta **dentro** de un proceso.
- Proceso y hebra son conceptos diferentes y suelen ser tratados por separado.
  - La función de un proceso es **agrupar recursos**.
  - Las hebras son **entidades planificables** para su ejecución sobre un procesador.
- Las hebras añaden a los procesos la capacidad de ejecutar varios conjuntos de instrucciones de forma concurrente dentro de un mismo entorno de procesamiento.
- Ejecutar varias hebras en paralelo es parecido a ejecutar varios procesos salvo por que comparten su recursos.
- A veces son llamados **procesos ligeros** o subprocesos.

Independencia/Cooperación

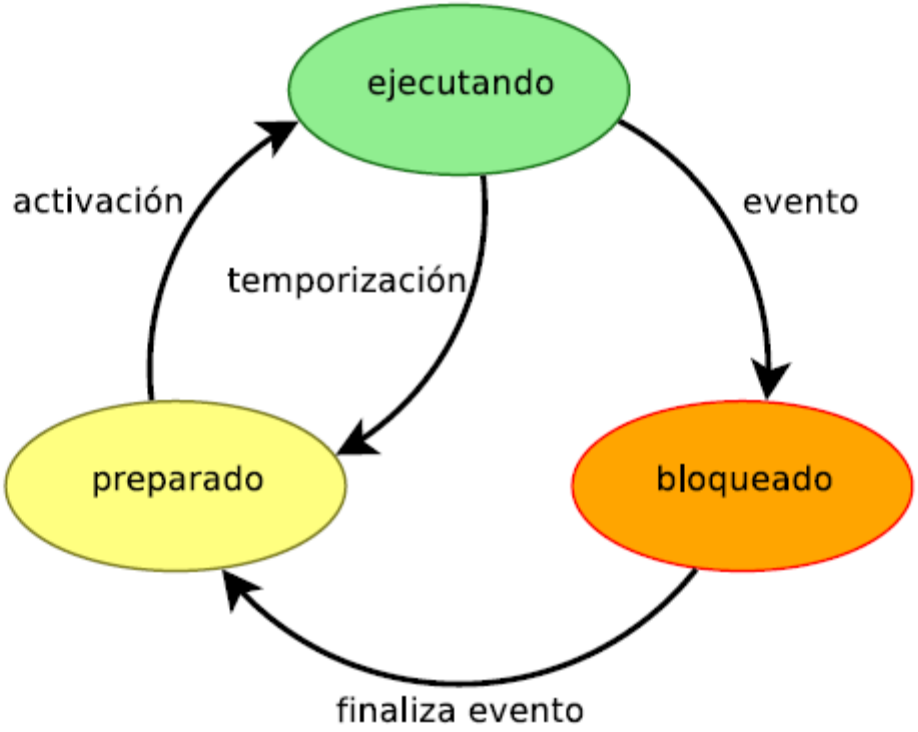
- Las diferentes hebras de un proceso **no** son **independientes** como las que se encuentran en diferentes procesos.
- Todas las hebras **comparten** los mismos **recursos**: espacio de direcciones, variables globales, ficheros abiertos,...
- Una hebra puede **alterar** cualquier parte de otra con la que comparta proceso.
- **No** hay **protección** entre hebras porque...
  - **es imposible**: comparten recursos.
  - **no es necesario**: diferentes procesos pueden pertenecer a diferentes usuarios, en cambio, todas las hebras de un proceso pertenecen al mismo.

Recursos necesarios

por proceso	**por hebra**
espacio de direcciones	identificador
variables globales	registros
ficheros abiertos	pila
procesos hijos	estado
alarmas pendientes	datos privados
señales y manejadores de señales	
información contable	
...	

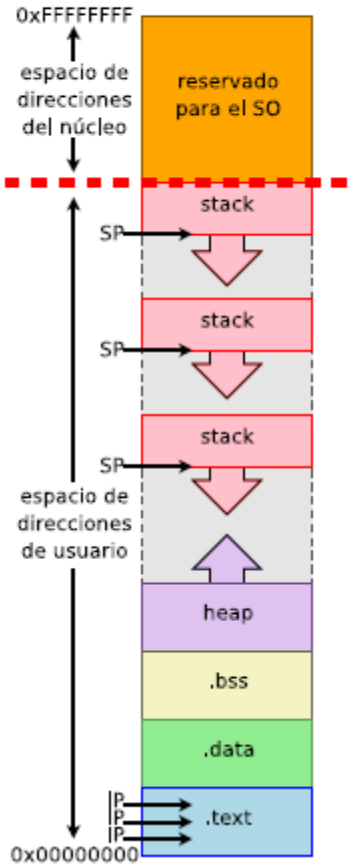
Estado de un hebra

- Al igual que los procesos, una hebra puede pasar por diferentes estados a lo largo de su ejecución.



Espacio de direcciones

- Cada hebra necesita su **propia pila**.
- Una pila contiene un **marco** de pila **por** cada **procedimento** llamado pero que no ha retornado todavia.
- Si en un hebra ejecutamos el procedimiento `X`, este llama al `Y`, y este llama `Z`, durante la ejecución de `Z` la pila contendrá los marcos de `X`, `Y` y `Z`.



Motivos para el uso de hebras

- **Compartición** de recursos, ej: servidor web.
- **Facilidad** (economía) de creación y destrucción.
- Mejora del **rendimiento**:
  - gran mejora en los limitados por E/S.
  - pequeña mejora en los limitados por procesador.
- En sistemas con múltiples procesadores es posible un verdadero **paralelismo**.
- **Simplicidad** de programación (a veces), ej: procesador de texto.

Ejemplos de uso de hebras

- Procesador de textos: utilizará una hebra por cada tarea que realiza.
  - Interacción con el usuario.
  - Corrector ortográfico/gramatical.
  - Guardar automaticamente y/o en segundo plano.
  - Mostrar resultado final en pantalla (`WYSIWYG`).
- Hoja de cálculo:
  - Interacción con el usuario.
  - Actualización en segundo plano.
- Servidor web: dos tipos de hebras.
  - Interacción con los clientes (navegadores).
  - Gestión del caché de páginas.
- Navegador: varios tipos de hebras.
  - Interacción con el usuario.
  - Interacción con los servidores (web, ftp, ...).
  - Dibujo de la página (1 hebra por pestaña).

Representación de hebras

- Para poder gestionar hebras necesitamos estrucuras de datos que las representen $\Longrightarrow$ **bloque de control de hebra** (**TCB**).

TCB mínimo
identificador de hebra (TID)
puntero de instrucción (IP)
puntero de pila (SP)
estado (flags)

Atributos de una hebra

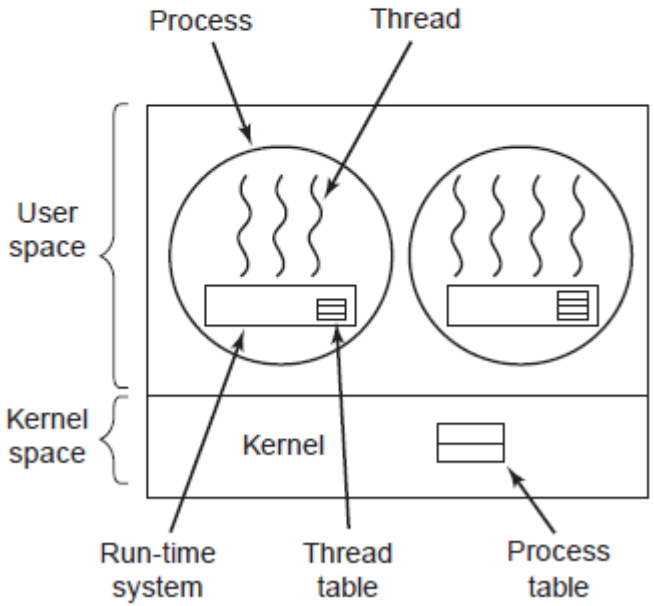
- Identificador de hebra (TID).
- Contexto:
  - Registros de usuario.
- Planificación.
- Pila.
- Recursos privados (opcional):
  - Datos privados.
- Otros:
  - Eventos relacionados: esperando E/S.
  - Prioridades.
  - Comunicación entre procesos.
  - Información de mapeado.
  - Propiedad y utilización de recursos.

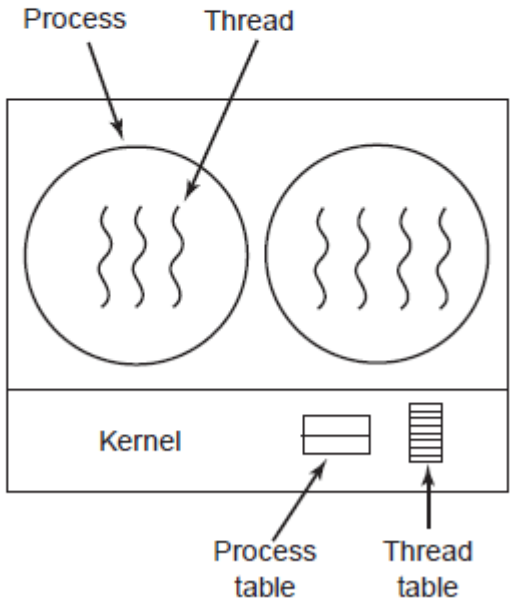
Implementación de los bloques de control de hebra

- ¿Dónde colocar los TCBs?
  - No colocarlos aleatoriamente.
  - Tenga en cuenta la TLB y la caché.
  - Diseñarlos con sumo cuidado (tanto o más que los PCBs).
- Posibles estructuras de datos:
  - Contiguas:
    - vector
    - vector de punteros.
  - No contiguas:
    - lista.
- Posibles localizaciones de los TCBs.
  - Espacio de usuario.
  - Espacio del núcleo.
  - Repartido entre ambos: TCB = UTCB + KTCB.

Tipos de hebras

- Hebras tipo usuario.
- Hebras tipo núcleo.
- Implementaciones híbridas.





Hebras tipo usuario

- El sistema de hebras se coloca por completo en el espacio de usuario.
- El núcleo no sabe nada de ellas y por lo que a él respecta se dedica a planificar procesos ordinarios.
- Todos los SO comenzaron perteneciendo a este tipo.
  - la mayoría ha evolucionado hacia el uso de hebras tipo núcleo.
- Cada proceso necesita una tabla de hebras.
  - Análoga a las tablas de procesos.
  - Gestionada por el sistema de ejecución de hebras.

Contenido de una biblioteca de hebras

- Creación y destrucción de hebras.
- Paso de mensajes entre hebras.
- Planificación de hebras.
- Sincronización de hebras.
- Almacenamiento y restauración del estado de una hebra.

Bibliotecas de hebras

- Pthreads (POSIX threads/estándar IEEE 1003.1c).
  - API para la creación y sincronización de hebras.
  - Especificación $\neq$ implementación.
  - Disponible en la mayoría de los SOs.
- Hebras Win32.
- Hebras Java.

Pthreads: API

```
#include <pthread.h>
```

función	**descripción**
`pthread_create(id, attr, func, arg)`	crea una nueva mediante la ejecución de la función `func`
`pthread_exit(val)`	finaliza un hebra y devuelve un valor para otra que quiera esperarla
`pthread_join(id, val)`	espera una hebra que finaliza y recupera el valor que devuelve
`pthread_self()`	devuelve el identificador de la hebra con tipo `pthread_t`
`pthread_yield()`	ceder el procesador voluntariamente

Ventajas de las hebras tipo usuario

- Pueden implementarse sobre **cualquier SO**.
- La **conmutación** entre hebras es varios órdenes de magnitud más **rápida** que la de procesos.
- Cuando una hebra termina de ejecutarse (`thread_exit()` / `thread_yield()`) puede cambiarse a otra **sin involucrar al núcleo** con lo que ahorraremos uno o dos cambios de proceso y modo.
- Cada proceso puede utilizar un algoritmo de **planificación personalizado**.

operación	hebras usuario	procesos
proceso nulo	34	11300
signal/wait	37	1840

Inconvenientes de las hebras usuario

- **Llamadas al sistema bloqueantes**: detener una hebra implica detener el proceso completo, ej: lectura de teclado, fallo de página. Soluciones:
  - modificar el SO para hacer sus llamadas no bloqueantes $\Rightarrow$ no queremos/podemos hacerlo.
  - comprobar si las llamadas bloqueantes tendrán éxito antes de efectuarlas, ej: `select/read` $\Rightarrow$ método engorroso.
- Como no existe interrupción de reloj no podremos ejecutar una hebra hasta que otra deje libre el procesador **voluntariamente** (cooperación obligatoria).
  - solución: solicitar interrupción periódica $\Rightarrow$ pérdida de rendimiento.
- Las hebras mejoran el rendimiento de tareas que se bloquean a menudo, pero una vez en modo núcleo es casi igual de costoso cambiar de proceso que volver al proceso original.
- En sistemas multiprocesador no podemos asignar más de un procesador a cada proceso.

Ventajas e inconvenientes de las hebras de usuario

Ventajas:	**Inconvenientes:**
la gestión de las hebras no requiere del núcleo ⇒ no requiere cambio de modo ⇒ mayor **velocidad**	si una hebra se bloquea ⇒ se **bloquean** todas las hebras del proceso
la política de planificación puede ser específica por aplicación ⇒ utilizar la que mejor **convenga**	el núcleo sólo puede asignar el procesador a procesos ⇒ 2 hebras del mismo proceso nunca podrán ejecutarse simultáneamente ⇒ sólo **concurrencia**
las hebras de usuario pueden utilizarse en **cualquier SO** (si dispone de una biblioteca de hebras)	

Hebras tipo núcleo

- El núcleo conoce la existencia de las hebras y se encarga de gestionarlas.
- No se necesita un sistema de gestión de hebras ni tablas de hebras en el interior de cada proceso.
- En el interior del núcleo ya disponemos de las tablas de procesos, las tablas de hebras en realidad son parte de ellas (replicando ciertas partes) $\Rightarrow$ algunos sistemas (Linux) no distinguen entre procesos y hebras.
- La **gestión** de hebras se hace a través de **llamadas al sistema** $\Rightarrow$ mayor coste de operación.

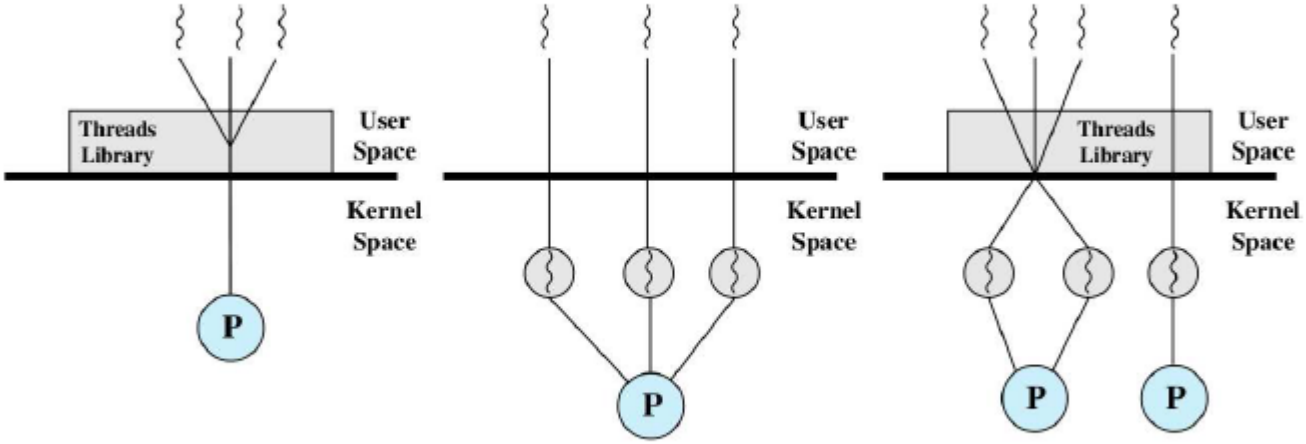
Ventajas e inconvenientes de las hebras del núcleo

- Ventajas:
  - No requieren llamadas al sistema **no bloqueantes**.
  - Si una hebra se bloquea o excede su tiempo de ejecución otra puede ser ejecutada.
  - En sistemas multiprocesador tantas hebras de un mismo proceso como procesadores existan pueden ejecutarse en **paralelo**.
- Inconvenientes:
  - **Coste** de operación mucho mayor que hebras tipo usuario.
  - Se tiende al **reciclaje** de hebras: finalizado $\Rightarrow$ nuevo.

En resumen...

- Hebras tipo usuario:
  - Rápidas y fáciles de implementar.
  - No planificables por el núcleo.
  - Se bloquean con E/S.
  - Sólo pueden usar un procesador.
  - Requieren cooperación entre hebras.
- Hebras tipo núcleo:
  - Involucran al SO, fraccionamiento del tiempo.
  - Operaciones de cambio y sincronización más costosas.
  - No se bloquean con E/S.
  - Pueden utilizar varios procesadores.
- Hebras híbridas:
  - Lo mejor de cada tipo.
  - Requieren balanceo de carga: reparto de hebras de usuario sobre hebras del núcleo para repartir el trabajo entre los procesadores.

Comparativa entre hebras y procesos



(a) Pure user-level (b) Pure kernel-level (c) Combined

Configuraciones en función del número y tipos de hilos soportados

- Múltiples hilos de usuario sin soporte de hilos de núcleo
  - Si una llamada al sistema requiere bloqueo del hilo, bloquea el proceso completo
- Un hilo de núcleo por cada hilo de usuario
  - No es necesario la biblioteca de hilos
  - Si un hilo se bloquea por una llamada al sistema, puede planificar otro hilo del mismo proceso
- Menor número de hilos de núcleo que hilos de usuario

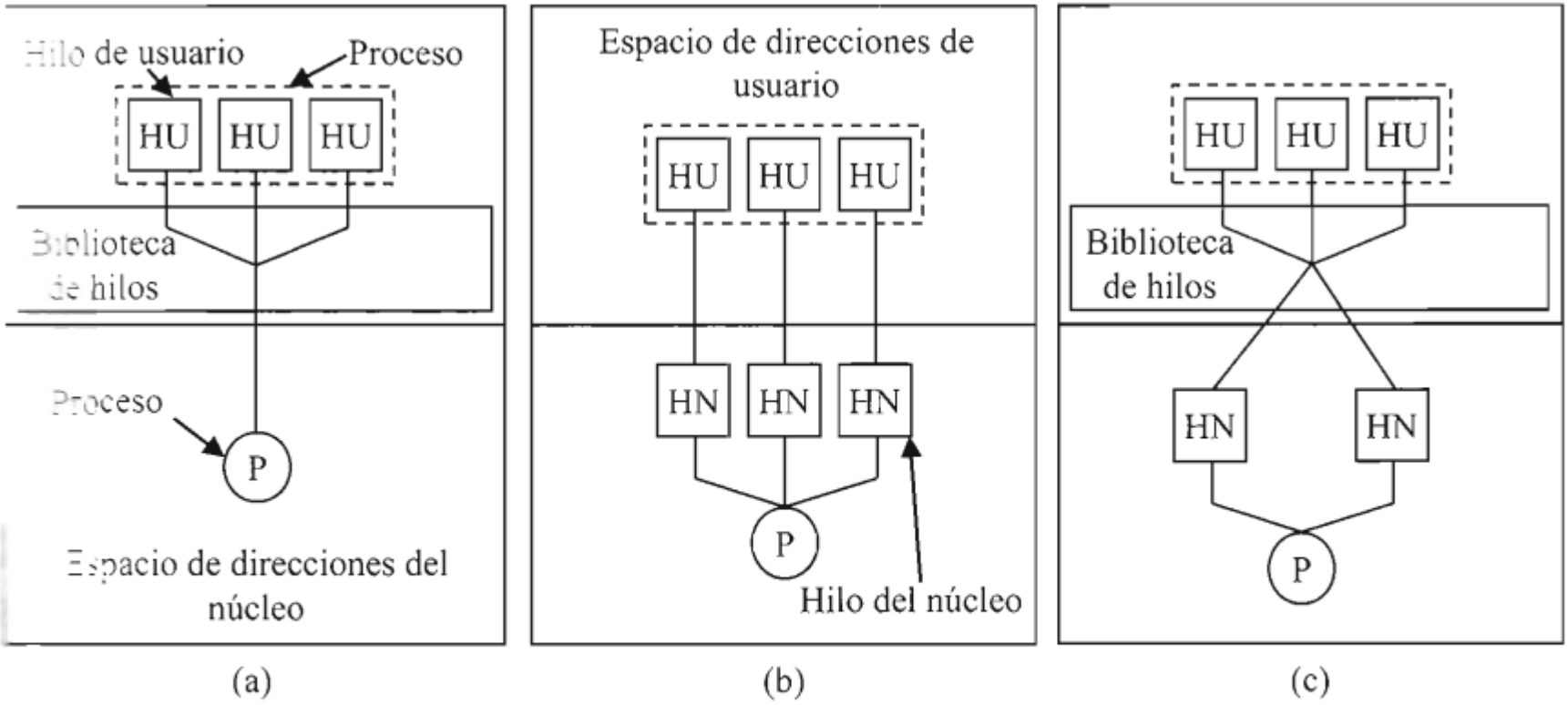


Figura 2.8 – Principales configuraciones en función del número y tipo de hilos soportados por un sistema operativo: múltiples hilos de usuario sin soporte de hilos del núcleo (a), un hilo del núcleo por cada hilo de usuario (b), y menor número de hilos del núcleo que hilos de usuario (c)

3. *(2 p)* Enumerar y describir **brevemente** las capas de software de E/S del núcleo de un sistema operativo.

Se denomina *cambio de proceso* o *cambio de contexto* a la operación que realiza el sistema operativo consistente en interrumpir la ejecución de un proceso A para iniciar o continuar la ejecución de otro proceso B. Entre las principales causas que motivan un cambio de contexto se encuentran las siguientes:

- *El proceso en ejecución pasa al estado bloqueado*. Un proceso A pasa al estado bloqueado cuando debe esperar por la aparición de algún evento. Mientras llega a producirse dicho evento se puede ejecutar otro proceso B.
- *La terminación (voluntaria o forzada) del proceso en ejecución*. Obviamente si un proceso termina se puede pasar a ejecutar otro.
- *El sistema operativo termina de atender una interrupción y existe un proceso B en estado preparado de mayor prioridad que el proceso actual A*. En dicho caso dependiendo de la política de planificación que siga el sistema operativo, se puede ejecutar dicho proceso B más prioritario o continuar con el proceso A que se estaba ejecutando.
- *El proceso A en ejecución ha excedido el tiempo máximo de ejecución ininterrumpida*. En los sistemas de tiempo compartido, a cada proceso se le asigna un tiempo máximo de ejecución ininterrumpida con objeto de poder atender las peticiones de todos los usuarios conectados al sistema. Superado dicho tiempo se produce un cambio de proceso, el proceso A en ejecución pasa al estado preparado y un proceso B en dicho estado pasa al estado ejecutándose.

2. *(2 p)* Enumerar las acciones que debe realizar el sistema operativo para crear un proceso.

El sistema operativo es el responsable de la creación de los procesos que se van a ejecutar. Las acciones que se deben realizar para crear un proceso dependen de cada sistema operativo, aunque de forma general se pueden distinguir las siguientes:

1. *Comprobar si el proceso puede ser creado*. El sistema operativo comprueba si existe suficiente espacio en memoria principal para crear otro proceso y si el usuario no ha excedido el número máximo de procesos que puede tener ejecutándose. Algunos sistemas operativos establecen un límite al número máximo de procesos que un usuario puede estar ejecutando para evitar exceder la capacidad de la tabla de procesos.
2. *Asignar una entrada de la tabla de procesos para el nuevo proceso*. En este instante se le asigna un identificador numérico al proceso.
3. *Reservar espacio en memoria para el proceso*. Se debe reservar espacio en memoria principal para el espacio de direcciones de memoria lógica del proceso y para la información de control del proceso.
4. *Inicializar el bloque de control del proceso*. Los campos de la entrada asociada al proceso en la tabla de procesos se inicializan con los valores adecuados.
5. *Establecer los enlaces apropiados*. El sistema operativo debe configurar los punteros adecuados para enlazar la información del proceso en las diferentes listas, tablas y colas que mantiene. Por ejemplo, el sistema operativo suele colocar al proceso creado en la cola de procesos preparados para que así puede ser planificado.

1. Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:
I) (*1 p*) Una de las principales ventajas de usar hilos del núcleo es que reducen la sobrecarga del sistema.
II) (*1 p*) La realización de un cambio de hilo a nivel de usuario implica la realización de un cambio de modo.

Solución Ejercicio 1

I) El principal inconveniente de los hilos del núcleo radica en que su gestión contribuye a la sobrecarga del núcleo. Para resolver este problema, muchos sistemas limitan el número de hilos del núcleo que se pueden crear, además reciclan los hilos del núcleo ya existentes. En conclusión la afirmación es **FALSA**.

II) La realización del cambio de hilo a nivel de usuario dentro de un mismo proceso se realiza sin necesidad de realizar un cambio de modo y un cambio de contexto. En conclusión la afirmación es **FALSA**.

3. (*2 p*) Explique **razonadamente** las características de los principales tipos de estructuras que puede presentar el núcleo de un sistema operativo.

Las características de los principales tipos de estructuras que puede presentar el núcleo de un sistema operativo son:

- *Estructura monolítica o simple*. Se caracteriza porque todos los subsistemas y las estructuras de datos del núcleo están ubicadas en un único módulo lógico, no existiendo interfaces bien definidos entre los subsistemas. El software del núcleo está escrito como un conjunto de procedimientos. Hay un procedimiento principal, un conjunto de procedimientos de servicio y un conjunto de procedimientos auxiliares. Típicamente el procedimiento principal invoca al procedimiento de servicio requerido por una llamada al sistema. A su vez el procedimiento de servicio invoca a los procedimientos auxiliares que necesita. Un procedimiento es visible por todos los demás.

- *Estructura en módulos* o *modular*. Se caracteriza por la existencia de varios módulos que pueden contener uno o varios subsistemas. Tanto los módulos como los subsistemas que contienen tienen una interfaz bien definida en términos de entradas, salidas y funciones que realizan. Además se pueden implementar como tipos de datos y objetos abstractos.

- *Estructura en capas* o *niveles*. Se caracteriza porque el núcleo estar organizado en una jerarquía de capas, cada una de las cuales subyace sobre la anterior. Cada capa i se implementa como un objeto abstracto que encapsula una serie de estructuras de datos y la implementación de las operaciones que pueden manipularlas. Dichas operaciones pueden ser invocadas por las capas de mayor nivel $i + 1$, $i + 2$,... Asimismo la capa i puede invocar a las operaciones de las capas de los niveles inferiores $i - 1$, $i - 2$,...

- *Estructura extensible*. Se puede considerar como un caso especial de la estructura modular. Se caracteriza por la existencia de un módulo esencial denominado *núcleo extensible* o *micronúcleo* y varios módulos accesorios denominados *extensiones del núcleo*. El micronúcleo se ejecuta en modo núcleo y se encarga de realizar únicamente los servicios absolutamente esenciales del sistema operativo. Los servicios menos esenciales del sistema operativo se implementan como extensiones del núcleo y se ejecutan en modo usuario.