Tema 4
Concepto de proceso

4.1. Definición

4.2. Estados de un proceso

4.3. Transiciones de estado de proceso

4.4. El descriptor de proceso

4.5. Gestión de procesos en Unix

Última modificación 07/03/13

E.T.S.I. Telecomunicaciones
José Luis Triviño Rodriguez

Definición

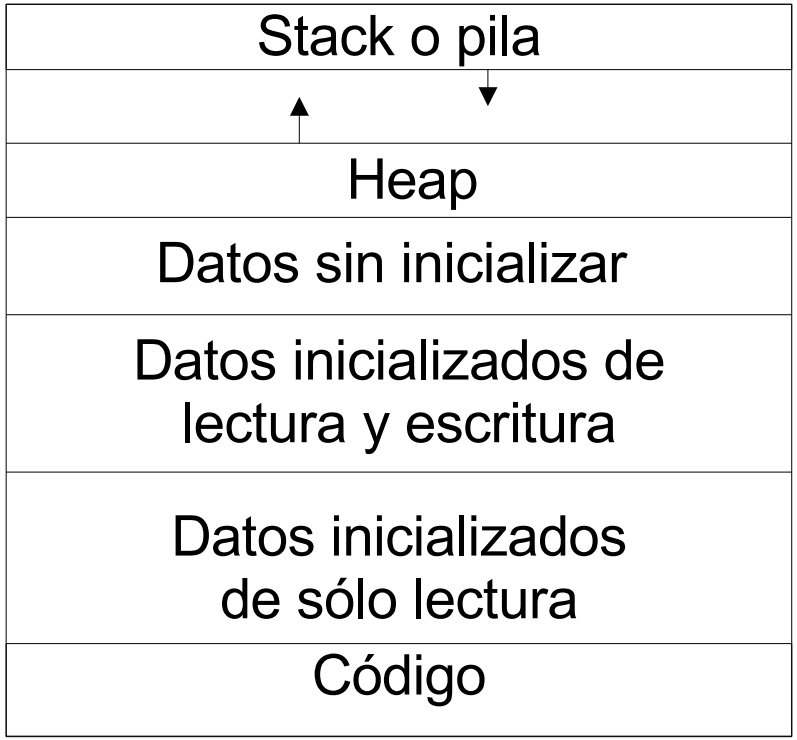
- El concepto de proceso es fundamental para la estructura de un SO, ya que éste consiste básicamente en una colección de procesos
- También se denominan tareas o trabajos
- Concepto utilizado por primera vez en la definición del SO Multics

Definición

- Definiciones de proceso
  - Programa en ejecución
  - Actividad asíncrona
  - “Espíritu animado” de un procedimiento
  - “Centro de control” de un procedimiento en ejecución
  - Lo que se manifiesta por la existencia de un “bloque de control de proceso” en un SO
  - Entidad a la que se asigna un procesador
  - Unidad “despachable”

Definición

- Espacio de direcciones de un proceso



Última modificación 07/03/13

Estados de un proceso

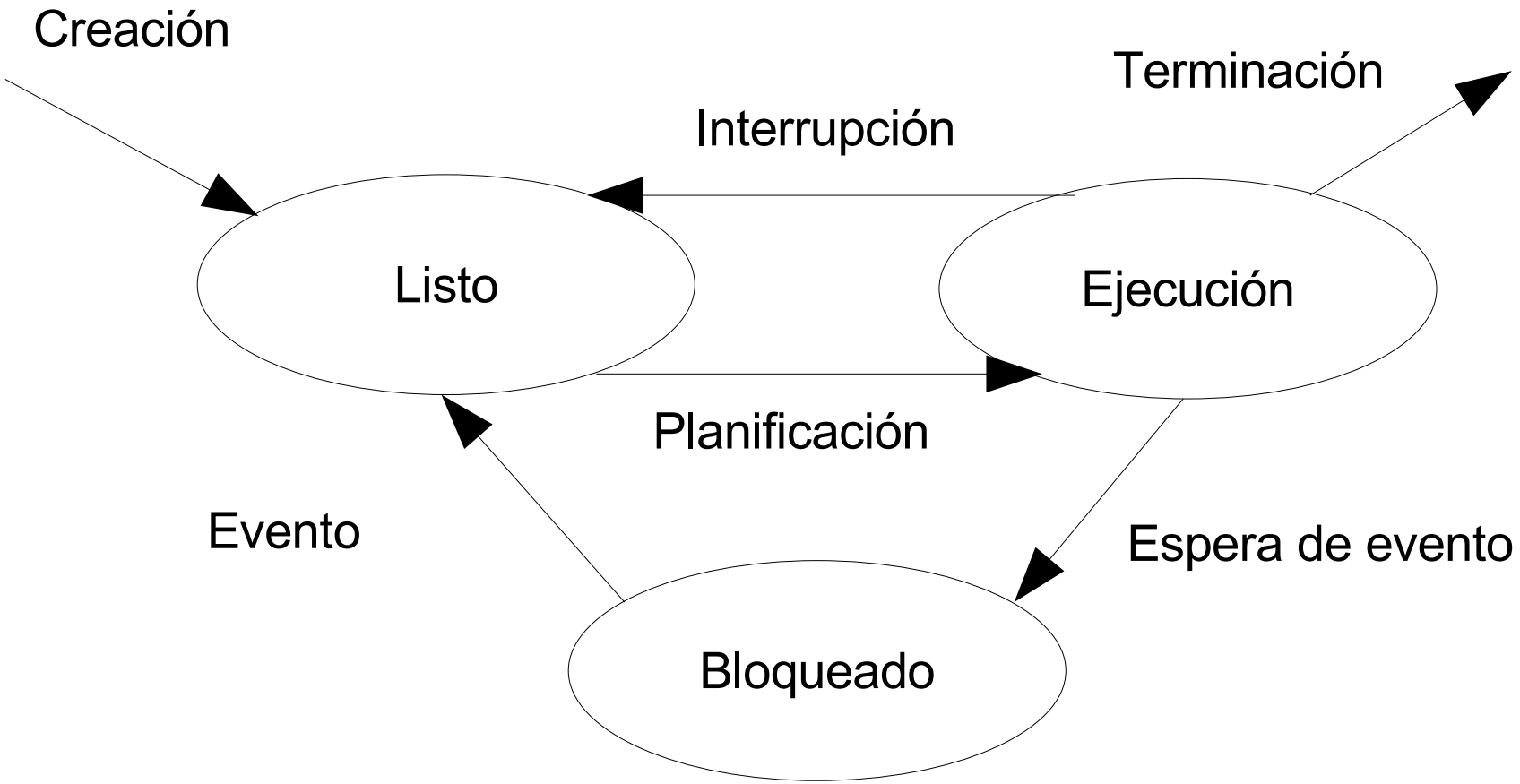
- El estado de un proceso es la disponibilidad para ser ejecutado de un proceso
- Un proceso, a lo largo de su ejecución, pasa por varios estados
- El cambio de un estado a otro es provocado por la ocurrencia de un evento

Estados de un proceso

- Los estados en los que puede encontrarse un proceso son:
  - En ejecución
  - Listo
  - Bloqueado

Última modificación 07/03/13

Transiciones de estado



El descriptor de proceso

- También se llama *Bloque de Control de Proceso* o *Vector de estado*
- Es una estructura de datos gestionada por el núcleo del SO
- Representa físicamente un proceso
- Todos los procesos tienen un descriptor de proceso
- Pueden existir descriptores de procesos que no tengan procesos asociados

El descriptor de proceso

- Información almacenada en el PCB:
  - Estado actual del proceso
  - Identificador único del proceso
  - Apuntador hacia el padre del proceso
  - Apuntador a los hijos del proceso
  - Información para planificación
  - Apuntadores a las zonas de memoria del proceso
  - Apuntadores a los recursos asignados
  - Área de salvaguarda de registros
  - Procesador

Gestión de procesos en Unix

- En términos prácticos, un proceso en Unix es una entidad creada mediante la llamada al sistema *fork*
- Todo proceso excepto el proceso 0 se crea cuando otro proceso ejecuta la llamada al sistema *fork*
- El proceso 0 es un proceso especial creado cuando el kernel inicia su ejecución

Gestión de procesos en Unix

- Tras crear dos procesos hijos (el proceso 1 y 2) el proceso 0 se convierte en el proceso *swapper*
- El proceso 1 se denomina *init* y es el antepasado de todos los demás procesos del sistema
- El proceso 2 es el *page daemon*

Estructuras para el control de procesos

- Unix mantiene una serie de estructuras de datos para el control de procesos:
  - Tabla de procesos
  - Área de usuarios
  - Tabla de regiones por proceso
  - Tabla de regiones
- Cada proceso dispone de dos pilas de datos: una de usuario y otra de kernel

Tabla de procesos

- Contiene los bloques de control de proceso (PCB) de cada proceso
- Cada PCB contiene:
  - El PID del proceso y el PID de su proceso padre (PPID)
  - El identificador del usuario read (UID), del efectivo y del grupo (GID)
  - El estado del proceso
  - Puntero a la tabla de regiones del proceso
  - Información de señales
  - Parámetros de planificación

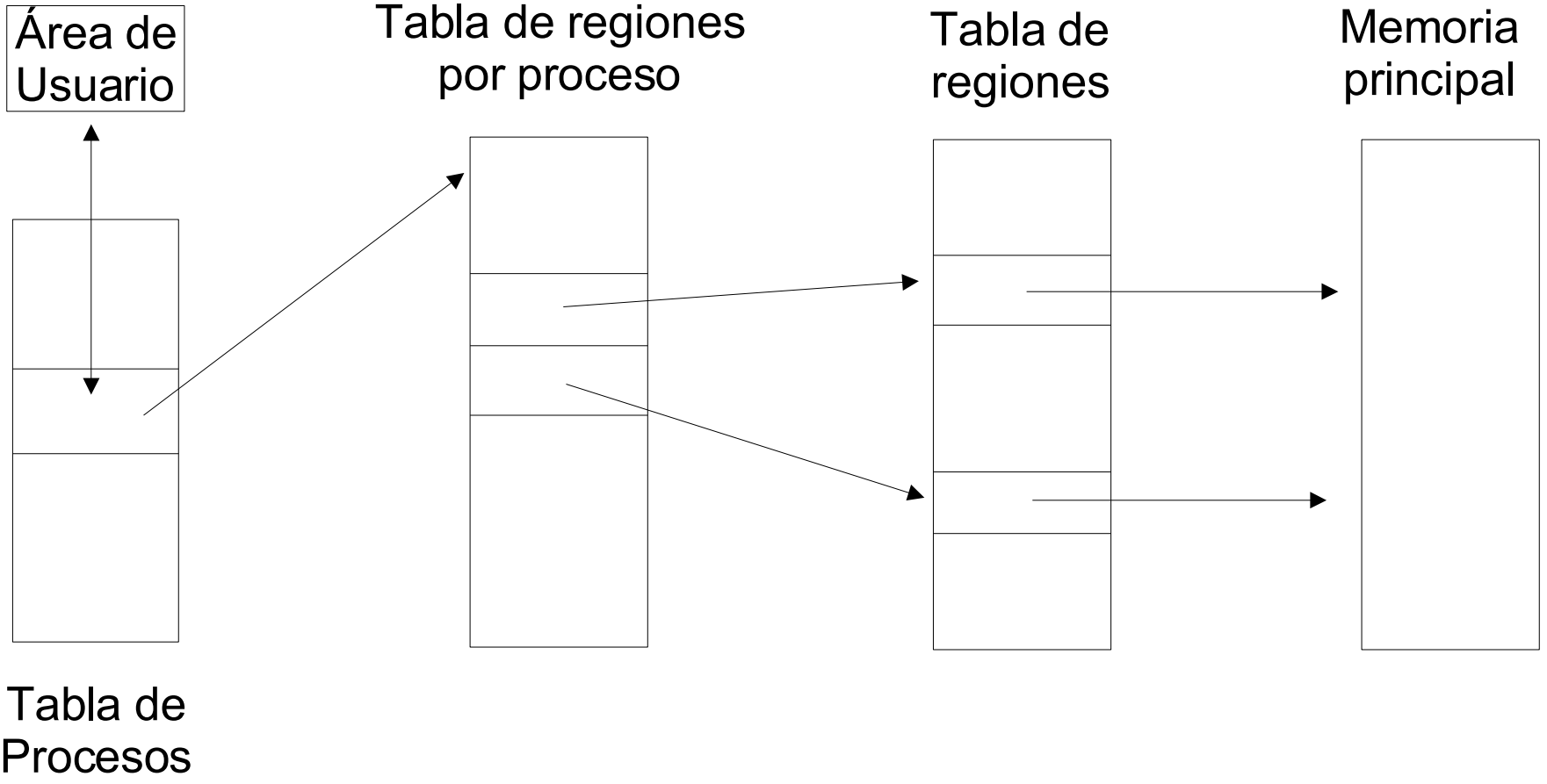
Área de Usuario

- Contiene información necesario únicamente cuando el proceso está en ejecución
- Un proceso puede acceder directamente a su área de usuario pero no a la de otros procesos
- Contiene:
  - Puntero a la tabla de procesos
  - Parámetros de llamadas al sistema
  - Tabla de descriptores de ficheros
  - Carpeta actual y carpeta raiz

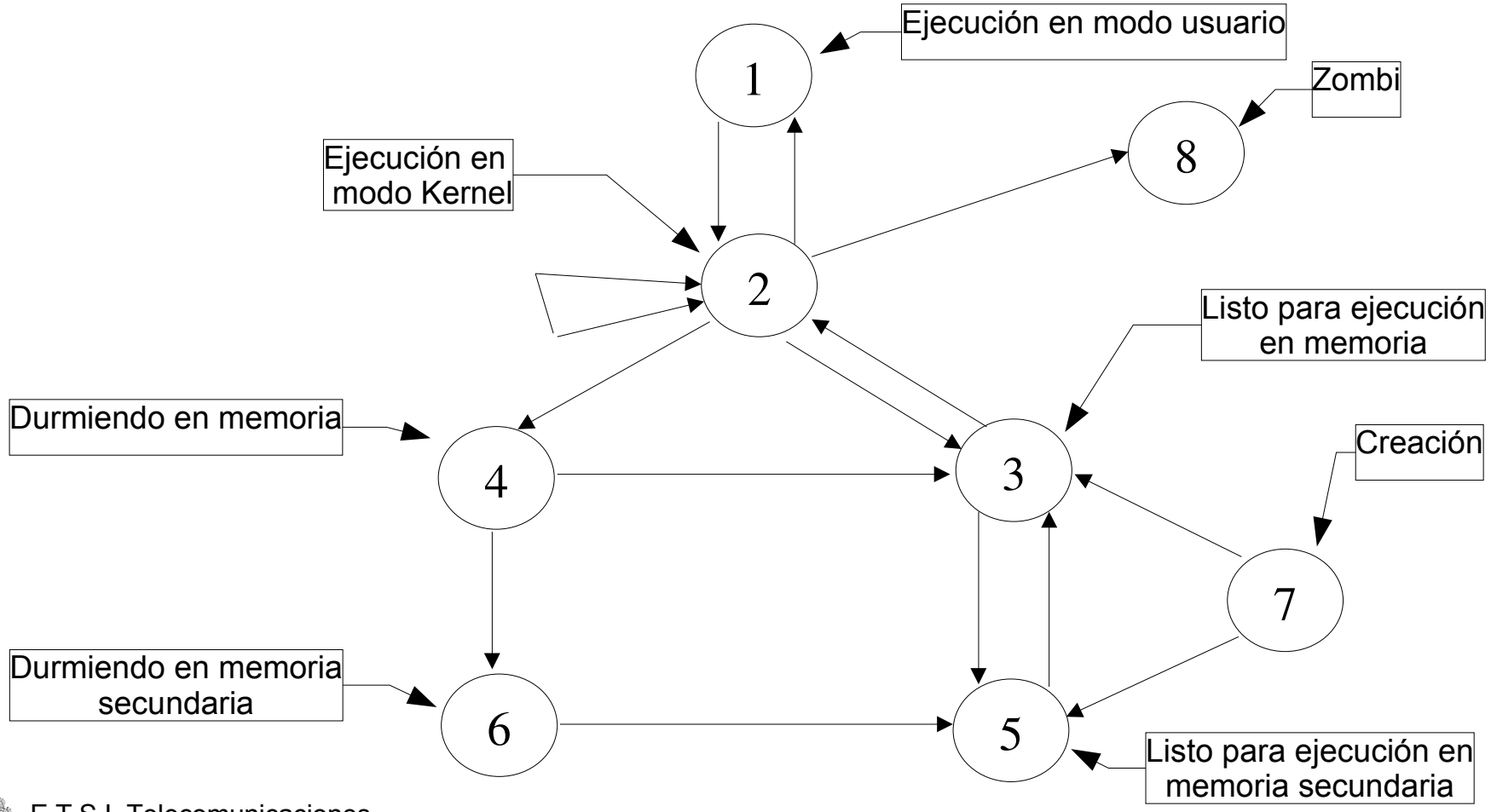
Tabla de regiones por proceso

- La tabla de regiones por proceso contiene una entrada por cada región del proceso
- Una región es una zona contigua del espacio de direcciones del proceso
- Cada región puede contener código, datos o pila
- Combinada con la tabla de regiones permite compartir regiones de memoria entre procesos

Tabla de regiones por proceso



Estados de un proceso



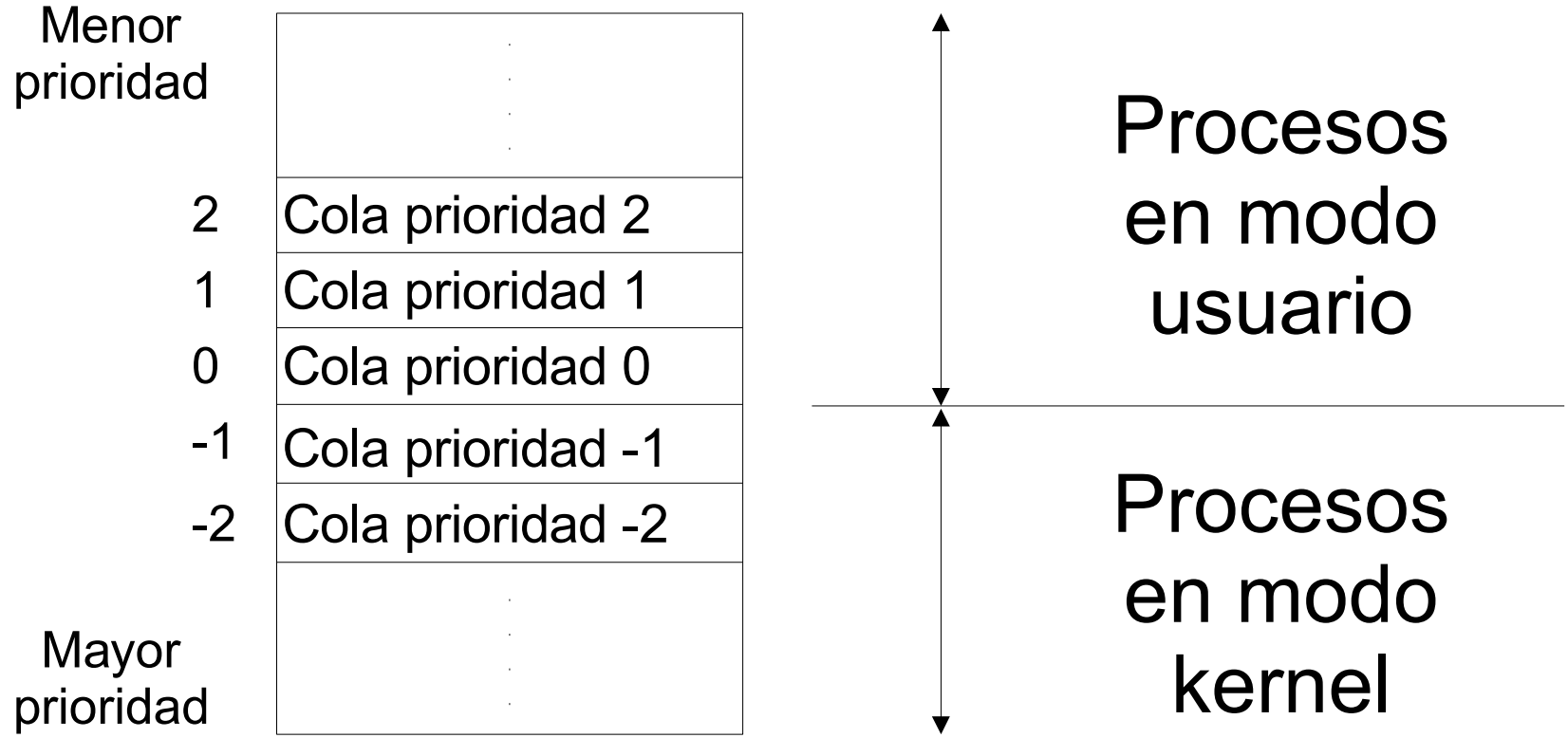
Planificación de procesos

- Unix utiliza un sistema de planificación round-robin con múltiples colas
- Cada cola tiene asociado un nivel de prioridad
- Los valores bajos indican alta prioridad y los altos menor prioridad
- Únicamente los procesos que están en memoria y listos para ejecución están en estas colas

Planificación de procesos

- Los procesos en modo usuario se encuentran por encima de un valor umbral y los procesos en modo kernel se ejecutan por debajo de este umbral
- A los procesos se les asigna un tiempo de CPU denominado *quantum* (habitualmente 100 ms)
- Un proceso se ejecuta hasta que agota su quantum o se bloquea voluntariamente

Planificación de procesos



Última modificación 07/03/13

Algoritmo de Planificación de procesos en Unix

- Cada segundo el planificador calcula la prioridad de cada proceso y se reorganizan las colas de prioridades
- Cada 100ms, el planificador selecciona el proceso de mayor prioridad y le asigna la CPU
- Si un proceso consume su quantum de tiempo, pierde la CPU y pasa al final de su cola de prioridades
- Si un proceso pasa al estado de dormido, el planificador le asigna la CPU al proceso más prioritario
- Tras una llamada al sistema, la CPU se asigna al proceso más prioritario
- Cada vez que se produce una interrupción de reloj (*tick* de reloj) se incremente el contador de CPU del proceso activo

Algoritmo de Planificación de procesos en Unix

- Los ticks de reloj es la forma de medir el tiempo del sistema operativo
- Varían desde 60 a 18 veces por segundo según el sistema
- En Unix se utilizan para calcular la prioridad de un proceso según la siguiente fórmula:

  prioridad = prioridad_base + (uso_de_CPU / 2)

- El 90% del uso de la CPU es olvidado por el algoritmo de planificación cada 5*n segundos (n = nº de procesos en el último min.)

Características de la Planificación de procesos en Unix

- Los algoritmos que utilizan E/S reciben un tratamiento favorecido
- Un proceso que consuma su quantum no será postergado indefinidamente porque el planificador olvida el uso de CPU
- Los procesos que usan CPU masivamente son ralentizados cuando la carga del sistema aumenta
- El sistema se ajusta a los cambios en la naturaleza del proceso

Ejercicio de Planificación de procesos en Unix

- Supongamos un sistema Unix donde hay 3 procesos A, B y C
- Todos los procesos han sido inicializados con una prioridad 0
- El sistema utiliza un rango de prioridades de usuario entre 0 y 20
- El tick de reloj es cada 1/60 segundos
- Tras 36ms, el proceso B realizará una E/S de 8ms
- Simular la planificación del sistema

Comunicación entre procesos

- Existen diferentes formas de comunicación entre los procesos en un sistema Unix:
  - Sockets
  - Semáforos
  - Señales
  - Tuberías (pipes)
  - Memoria compartida

Sockets

- Se desarrollaron por primera vez en la versión Unix BSD4.2
- Constituyen extremos de trayectorias de comunicación bidirecionales
- Se utilizan sobre todo para llevar a la práctica protocolos de red
- Permiten comunicar procesos localizados en diferentes máquinas a través de una red de transmisión de datos

Semáforos

- Se desarrollaron por primera vez en la versión Unix System V
- Permiten controlar el acceso a recursos compartidos
- Se utilizan para bloquear un recurso antes de usarlo
- Un nuevo intento de bloquear el recurso antes de su liberación provocaría que el nuevo proceso pasase a estado de espera

Señales

- Son mecanismos software similares a las interrupciones hardware
- Informan de forma asíncrona de la ocurrencia de un evento
- Las señales pueden ser generadas por:
  - El hardware
  - El núcleo
  - Procesos
  - Usuarios

Tuberías

- Constituyen canales de comunicación unidireccionales entre dos procesos
- Están formadas por una cola FIFO de bytes
- El sistema se encarga de forma automática de la sincronización y el manejo de buffers
- Unix System III introdujo las tuberías con nombre para permitir comunicaciones globales al sistema

Memoria compartida

- Es un mecanismo introducido en Unix System V
- Permite a varios procesos compartir bloques de memoria tanto para lectura como para escritura
- No incluye ningún mecanismo de sincronización
- Se basa en el mecanismo de memoria virtual del sistema y en la tabla de regiones