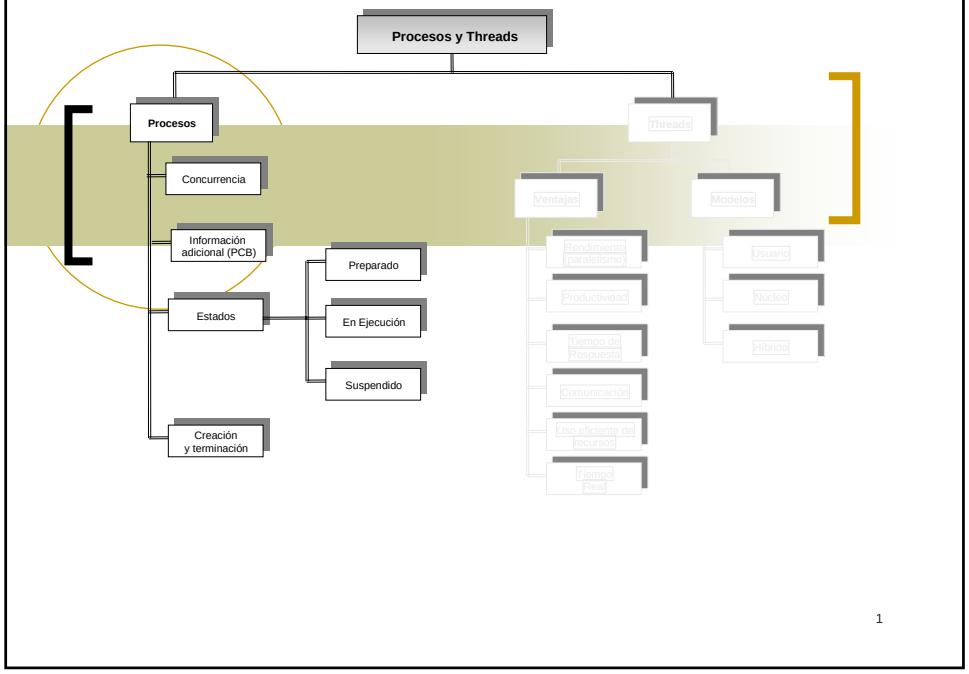


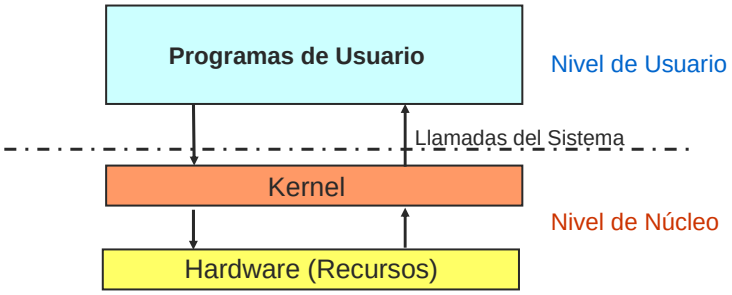
1

ÍNDICE

- Introducción: Sistema Operativos Multitarea
- Procesos:
  - Definición y propiedades
  - Estados de un proceso
  - Información de control del proceso
  - Creación y terminación de procesos
- Threads:
  - Definición y propiedades
  - Ventajas
  - Modelos de control de threads

Sistemas Operativos de Tiempo Real

- Estructura de un S.O. de Tiempo Real:
  - **Espacio de Usuario**:
    - Ejecuta las aplicaciones del usuario en su espacio de memoria
  - **Espacio de Núcleo:**
    - Ejecuta las tareas del **S.O.** de forma privilegiada accediendo a los recursos hardware del sistema



3

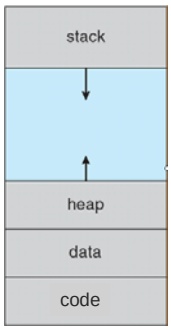
Sistemas Operativos Multitarea

- Un Sistema Operativo de Tiempo Real (SOTR) debe permitir la multitarea
- Multitarea: capacidad de ejecutar varias tareas concurrentemente (manteniéndolas en estados intermedios de ejecución al mismo tiempo)
- Mecanismos para implementar la multitarea:
  - **Paralelismo**: se dispone de varias CPUs y en cada una de ellas se ejecuta una tarea.
  - **Pseudoparalelismo**: el sistema solo dispone de una CPU y la concurrencia se consigue mediante periodos de tiempo de ejecución a cada tarea.
- El control y programación de sistemas multitarea es complicado.

Procesos: Definición y Propiedades (I)

- El concepto de **proceso** simplifica el control y programación del paralelismo
- Un proceso se puede ver como un programa en ejecución (¡OJO!, no es lo mimo que un programa)
  - **Programa**: Código ejecutable normalmente almacenado en un dispositivo secundario (por ejemplo un disco). Es una entidad PASIVA de cara al sistema.
  - **Proceso**: programa en ejecución -> con recursos del sistema (memoria, ficheros..) y que puede hacer uso de la CPU (**Planificable**)
  - Un programa se convierte en proceso una vez que se ha almacenado en memoria y el SO tiene constancia de él.

Procesos: Definición y Propiedades (II)

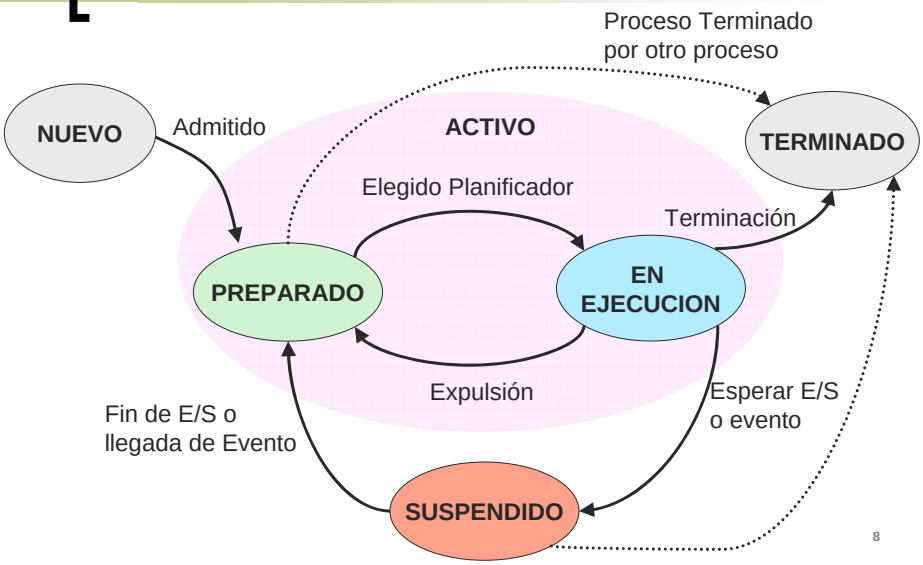


- Un proceso no consta sólo de código ejecutable, también tiene asociada información adicional para ser gestionado por el sistema. Entre otras:
  - Contador de programa
  - Registros de la CPU
  - Pila del proceso para datos temporales (parámetros de subrutinas, direcciones de retorno, variables temporales, …)
  - Sección de Datos (variables globales, …)
  - Información adicional para planificación
- Dos procesos se pueden asociar a un mismo programa (aunque son dos secuencias de ejecución distintas)
- Es habitual que un proceso genere varios procesos más durante su ejecución

Estados de un Proceso (I)

- Mientras un proceso está presente en un sistema, éste tiene un estado asociado
- El estado se define por la actividad que realiza dicho proceso
- Estados posibles:
  - **Preparado**: el proceso está esperando que se le asigne a un procesador para comenzar o seguir ejecutándose
  - **En Ejecución**: El proceso es "dueño" de un procesador y sus instrucciones se están ejecutando
  - **Suspendido**: el proceso espera que suceda algún evento (p.e. operación de E/S). Al suspenderse no degrada el rendimiento del sistema

Estados de un Proceso (II)



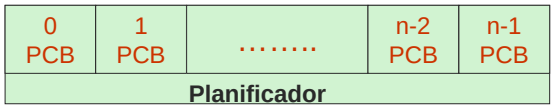
Estados de un Proceso (III)

- La transición entre estados no se realiza automáticamente, existe una parte del SO que se dedica a ello: ***el planificador de procesos***

- El planificador decide qué proceso se va a ejecutar en cada momento y con qué política de planificación (determina en gran medida el rendimiento del sistema)
- En un momento dado solo un proceso puede estar en ejecución en un procesador, pero puede haber varios en estado Preparado o Suspendido

Información Adicional sobre Procesos

- El SO almacena la información de los procesos en una *tabla de procesos*
- La tabla de procesos tiene una entrada por cada proceso



- Cada entrada de la tabla se denomina PCB (*Process Control Block*)
- EL PCB almacena toda la información referente al proceso: estado, PC, puntero a la pila, …
- Cuando se interrumpe la ejecución de un proceso y este pasa a estado *Preparado*, se guarda toda la información importante de forma que se pueda reanudar su ejecución como si no se hubiese detenido anteriormente

Bloque de Control de Proceso (PCB)

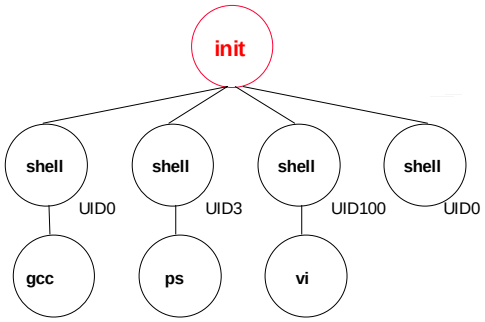
- El contenido del PCB varía de un SO a otro pero básicamente incluye la siguiente información:
  - Estado del proceso
  - Contador de programa (PC)
  - Registros de la CPU: acumuladores, registros índice, puntero a la pila, etc.
    - Esta información junto a PC permite continuar la ejecución cuando se produce la interrupción
  - Información de planificación de CPU (prioridades, punteros a colas de planificación, …)
  - Información de administración de memoria
  - Información contable (PID, ID usuario, tiempos, …)
  - Información de estado de E/S (solicitudes de E/S pendientes, dispositivos asignados, archivos abiertos, …)

Creación de Procesos (I)

- Un SO que permita la ejecución concurrente debe contar con un mecanismo de creación y terminación de procesos
- **Creación**: supone asignar todos los recursos que el proceso necesita para su ejecución (p.e. memoria, dispositivos E/S, etc…)
  - Durante la ejecución de un proceso puede crear procesos nuevos mediante una ***llamada al sistema***
  - Al proceso que crea se le denomina **PADRE** y a los creados **HIJOS**
  - Los hijos pueden obtener los recursos directamente del SO (UNIX) o compartir los del padre (evita la saturación del sistema)
  - *Ejemplo en UNIX*: **fork**()

Creación de Procesos (II)

- ¿Quién crea el proceso inicial?
  - init: es el proceso del SO (kernel) encargado de lanzar el resto de procesos del sistema y del usuario.
  - Este proceso se inicia durante el arranque (BOOT) del SO



Terminación de Procesos

- Un proceso termina cuando:
  - Concluye la última instrucción del código
  - Otro proceso solicita su terminación
- El SO (mediante una ***llamada del sistema***) elimina los procesos liberando los recursos asignados
- El proceso Padre debe esperar la terminación de los procesos Hijo antes de terminar él mismo.
  - La liberación de los recursos del proceso hijo queda bloqueada hasta que el padre *señaliza* (espera) su terminación
  - *Ejemplo en UNIX*: wait()