





TEMA 1 I 2: INTRODUCCIÓ I PROCESSOS

 Assign	
 Status	

Resumen para preparar el Exámen de Teoria 1 de SO.

Tema 1

1.1 Papel del S.O.

1.1.1 S.O.:

Software que controla los recursos disponibles del sistema y actúa de intermediario entre HW y Apps.

Internamente: Define estructuras de datos.

Externamente: Tiene funciones para acceder a sus servicios.

1.1.2 Qué ofrece el S.O.

Arranque:

- Carga S.O. adecuado en memoria (boot).
- Captura interrupciones.
- Inicializa acceso a sistema.

Uso:

- Ejecuta programas

1.2 Formas de acceder al Kernel

Por protección y seguridad, existen dos modos:

- User Mode: Low priority
- Kernel Mode: High priority

Ciertos accesos a memoria e instrucciones solo se permiten en Kernel Mode.

El código Kernel se ejecuta:

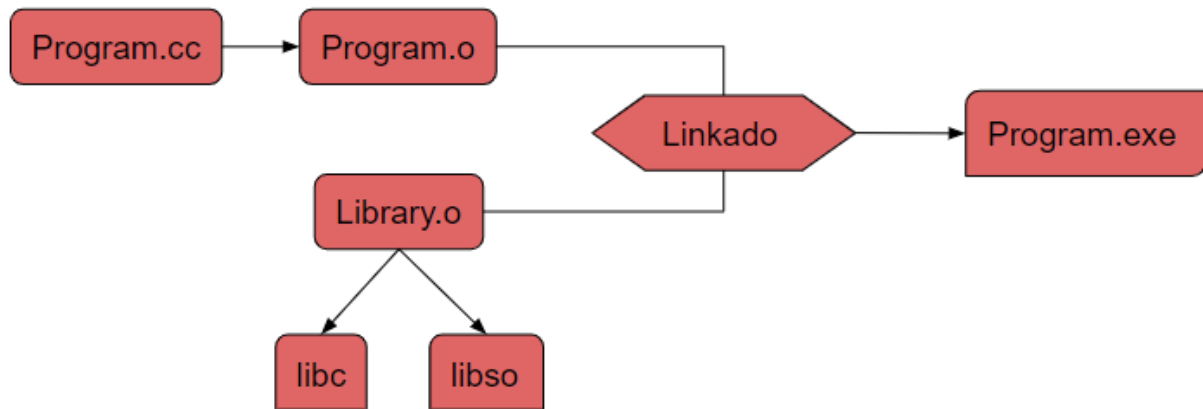
- Cuándo se llama a sistema: Peticiones de servicio voluntarias.
- Cuándo se provoca una excepción: Error de software (dividir entre 0, posición inválida de memoria, etc)
- Cuándo se provoca una interrupción: Es generada por el HW (teclado, reloj, etc)

Esta ejecución permite tener el control.

No obstante, si no se cumple ninguno de los 3 casos, el Kernel manda una interrupción de reloj cada cierto tiempo para evitar perder el control.

1.3 Llamadas a sistema

1.3.1 Generación de ejecutables



1.3.2 Definición de llamada

- Funciones para acceder a los servicios del Kernel.
- Para el programador debería ser una función más.

1.3.3 Requerimientos de llamadas

Para el programador, la llamada debe ser como una simple llamada a función. Y no se deben modificar los registros que se usen.

Para el Kernel, todas las direcciones de las llamadas tienen que ser variables, y deben soportar diferentes versiones del S.O.

Para solucionar esto, usamos una librería de sistema:

Traduce a función del usuario la llamada:

- Pasa parametros al Kernel
- Invoca al Kernel
- Recoge resultados del Kernel
- Generaliza (error siempre = -1)

Para mantenerse estable entre versiones, no se usa una posición de memoria sino un puntero a una HashTable que contiene la posición de memoria.

Tema 2

2.1 Conceptos

2.1.1 Concepto de procesos

Un proceso es un programa en ejecución. El ejecutable se divide en datos y código, y al ejecutarlo, se asigna memoria, se inicializan los registros CPU y se ofrece acceso al Kernel.

Toda la gestión de la información de un proceso se almacena en el PCB (Process Control Block).

La ejecución del programa genera un proceso.

2.1.2 PCB

Contiene la información para gestionar un proceso, desde la dirección de las regiones de datos, hasta el PID y toda la información sobre los procesos.

También contiene el valor del program counter.

2.1.3 Eficiencia

Si no contamos de CPUs de sobras, nos sale a cuenta crear procesos concurrentes para contar con ejecución en paralelo.

2.1.4 Hilos de ejecución - Threads

Un proceso es la unidad de asignación de recursos asignado a un ejecutable.

Los threads son los distintos hilos de CPUs que permite ejecutar por partes independientes.

Contiene:

- TID
- Puntero a pila
- Program counter (Puntero a la siguiente instrucción)
- Registros
- Variable err.no

Un proceso tiene siempre un thread asignado, pero puede tener más.

Pros

Intercambiar info sin llamadas a sistema (Shared Memory)

Contras

Difícil de programar y debuggar

Problemas de sincronización al compartir memoria

2.1.5 Estados de un proceso

Run	Ready	Blocked	Zombie
En ejecución	Preparado Espera asignación de CPU	No consume CPU Esta esperando a E/S o signal	Ha terminado pero no ha muerto

2.2 Servicios básicos para gestionar procesos

El sistema ofrece un conjunto de funciones para gestionar procesos

Creación → fork	Espera a hijo → waitpid
Mutación → execlp	Obtener PID → getpid
Terminación → exit	Obtener PID padre → getppid

2.3 Comunicación entre procesos

Para compartir información y acelerar la computación, los procesos necesitan comunicarse.

Se pueden comunicar mediante Shared Memory o Signals.

2.3.1 Signals

Forma de enviar información a través de eventos entre procesos.



2.4 Gestión interna de procesos

Estructuras de datos:

- Propiedades → PCB
- Threads → S.O.

Datos PCB:

- PID
- User, Group, etc.
- Estado (Run, Blocked, etc)
- Espacio para registros
- Datos para gestionar signals
- Información de planificación, memoria, E/S y recursos consumidos

2.5 Planificación

El S.O. organiza los PCB en estructuras de datos (por ejemplo, vectores)

Se suelen organizar por estados:

- Vector de procesos Ready
- Vector de procesos Running
- etc

El scheduler decide como organiza el tiempo de la CPU. El criterio en base al que decide se llama Política de Planificación.

El método de planificación se ejecuta en cada cambio en la ejecución:

- Si un proceso esta Running y termina, la política de planificación envía un nuevo proceso a usar la CPU y actualiza el vector de procesos Ready.

Existen dos tipos de políticas de planificación:

- Preemptivas: La política quita la CPU al proceso. Soporta eventos preemptivos y no preemptivos.
- No preemptivas: La politica no le quita la CPU, el proceso la libera. Sólo acepta eventos no preemptivos.

2.5.1 Round Robin (preemptiva)

Round Robin clasifica los procesos por estados, y los encola por orden de llegada.

Los procesos tienen un tiempo máximo de uso de CPU. Es conocido como quantum time.

Caso 1:

El proceso se bloquea → Se añade a la cola de bloqueados

Caso 2:

El proceso termina → Se pasa a la cola de zombies

Caso 3:

Proceso termina el quantum → Se añade al final de la cola de Ready.

2.5.2 Completely Fair Scheduling

Buscamos que el tiempo sea equivalente en todos los procesos.

En este caso, el tiempo máximo de uso de CPU viene definido por la siguiente fórmula:

$$t_q = \frac{t_p}{N_p}$$

Dónde t_q se refiere al tiempo de quantum.

Dónde t_p se refiere al tiempo que lleva el proceso usando la CPU.

Y dónde N_p se refiere al número de procesos pendientes de usar la CPU.

Prioridad → Distancia al tiempo teórico de CPU (t_q). A más tiempo, mayor prioridad tendría el proceso.

2.6 Seguridad

La seguridad debe considerarse a cuatro niveles distintos.

2.6.1 Físico

Las máquinas que controlan el S.O. y las terminales deben estar en un edificio asegurado, y en salas seguras.

2.6.2 Humano

Hay que controlar a quién se le da acceso a los sistemas y concienciar de no facilitar permisos a personas ajenas al sistema.

Es decir, los usuarios con permisos de acceso deben tener protegidos sus accesos en forma de contraseña.

2.6.3 Sistema Operativo

Hay que evitar que un proceso sature el sistema, que los servicios estan siempre operativos y que los puntos de acceso que no deseamos no esten operativos.

2.6.4 Red

Hay que evitar ataques de red, ya que la mayoría de datos hoy en dia se mueven por ese ámbito.