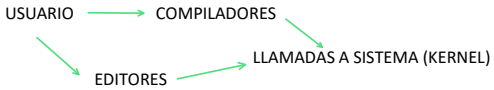
1. Introducción

martes, 7 de febrero de 2023 18:07

SO
-> Hace de intermediario.
-> Es un software que controla los recursos del hardware y actúa de intermediario entre aplicaciones y hardware, tiene privilegios de hardware.
-> Tiene muchas capas, cuanto más alta es la capa más complejidad tiene y más fácil es para nosotros (lo que se hace en 1 paso en frameworks son 15 en SO).

Internamente:	Define estructuras de datos para gestionar el hardware y algoritmos para decidir cómo usarlo
Externamente:	Ofrece un conjunto de funciones para acceder a su funcionalidad de gestión de recursos



LLAMADAS A SISTEMA
-> Provocar cambios en los datos del SO (ej: nuevo programa)
-> Nuevas acciones:
-Poner a ejecutar
-Ponerlo en la cola

COMPILAR EN C:
1. gedit p1.c
2. gcc -o p1 p1.c
3. p1

SO OFRECE:
1. Entorno usable: ayuda a que el usuario no vea diferencia entre sistemas
2. Entorno seguro: protege HW de accesos incorrectos, define permisos de acceso
3. Entorno eficiente: uso eficiente del sistema

PRESENTE DE INICIO A FINAL
Se ejecuta el código del SO cuando se hace llamada al sistema, excepción o interrupción involuntarios. El SO tiene interrupciones programadas porque si no se usaría y perdería el control, entonces las va provocando para asegurarse y comprobar el estado de la máquina.

¿QUÉ ES KERNEL?
Un kernel es el núcleo/corazón de un SO. Es el componente central que se encarga de administrar y coordinar los recursos del hardware (la CPU, memoria y dispositivos de E/S) para permitir que los programas y procesos del sistema funcionen de manera eficiente y sin interferirse entre sí.

- Se encarga de la gestión de memoria, procesos, gestión de interrupciones y gestión de dispositivos E/S.
- Proporciona una interfaz entre las aplicaciones y hardware subyacente al sistema.

Modo privilegiado(modo kernel/sistema/protegido): nivel de acceso más alto y está reservado para el kernel del SO y procesos de bajo nivel que necesitan acceso a recursos críticos del sistema (la memoria, dispositivos de E/S). En este modo los programas pueden ejecutar instrucciones que no puedes desde el no-priv (como las que permiten acceder directamente a los recursos de hardware).

Modo no-privilegiado(modo usuario): nivel de acceso más bajo, disponible para la mayoría de los programas y procesos que se ejecutan en el sistema(apps). Los programas tienen acceso limitado a los recursos del sistema.

FORMAS DE ACCEDER AL KERNEL
Kernel = núcleo del SO -> parte que se ejecuta en modo privilegiado.
La SO está formada por kernel+ aplicaciones (modo NO-privilegiado).

El código de kernel se ejecuta cuando:
- Hay llamadas a sistema (síncronas a nivel HW, voluntarias)
- Una aplicación provoca una excepción (síncronas, involuntarias)
- Una aplicación provoca una interrupción (asíncronas, involuntarias)

Si no ocurre ninguno de estos eventos, el SO no se ejecutaría -> perdería el control del sistema.
Para evitarlo el SO configura una interrupción automática cada "x" ms.

LLAMADAS A SISTEMA
Cuando el programa tiene que acceder al código del SO/Kernel/HW, las llamadas hacen el salto al código y cambian los "permisos".
Son un conjunto de funciones del kernel para acceder a sus servicios -> como cualquier otra librería.

Requisitos para LL.S (punto de vista kernel):
- Ejecución en modo privilegiado -> soporte HW
- Paso de parámetros y retorno de resultados entre modos de ejecución diferentes -> dependen del HW
- Las @ que ocupan LL.S tienen que poder ser variables -> PORTABILIDAD (tienen que poder soportar diferentes versiones del kernel y diferentes SO)

Requisitos para LL.S (punto de vista programador):
- Sencillo como una llamada a función
- No se puede modificar su contexto

*TRAP -> interrupción del SO causada por un error/requerimiento de usuario -> invoca al kernel.

¿Cómo se consigue la portabilidad? Las LL.S tienen un id(un número) no una @, este id se usa para indexar una tabla de LL.S que se conserva constante entre versiones

Librerías "de sistema" con soporte HW:
Se encarga de traducir la función que ve el user a la petición de servicio explícito del sistema.

- La librería de sistema aísla al user de los detalles de la arquitectura
- Modo privilegiado = seguridad

> Pasar los parámetros shell (kernel)
> Invocar el código del kernel (*TRAP)
> Recoger y procesar los resultados del kernel
> Homogeneiza resultados; ej: llamadas en linux que dan error siempre retornan -1)

Código genéricos al entrar/salir del kernel:

Hay pasos comunes a interrupciones, excepciones y llamadas a sistema.
En el caso de interrupciones y excepciones, no se invoca explícitamente ya que genera la invocación la realiza la CPU, el resto de pasos si se aplican.

	Función "NORMAL"	Función KERNEL
Pasamos los parámetros	Push parámetros	**DEPENDE**
Para invocarla	Call	Sysenter, int o similar
Al inicio	Salvar los registros que vamos a usar (push)	Salvamos todos los registros (push)
Acceso a parámetros	A través de la pila (ej: mov 8(ebp), eax)	**DEPENDE**
Antes de volver	Recupera los registros salvados al entrar (pop)	Recupera los registros salvados (todos) al entrar (pop)
Retorno de resultados	Eax (o registro equivalente)	**DEPENDE**
Para volver al código que la invocó	ret	Sysexit, iret o similar

GENERAR EJECUTABLE

- El ejecutable tiene que tener a parte de nuestro programa todas las librerías necesarias para que este funcione + las librerías de sistema que contienen los SYSCALLS