Clase 2 y 3

miércoles, 19 de marzo de 2025 18:17

Kernel:

- Administra memoria y gpu
- Escrito en c, assembler o rust
- Está en kernel/sched.c

Desarrollo colaborativo.
El kernel se divide en subsistemas. Cada uno lo mantiene distitas personas.
Linus Torvalds es el más pingudo, es quien en última instancia decide si aprueba o no el merge.
Monolítico híbrido:

- porque está todo en una misma imagen.
- Drivers y código del kernel se ejcutan en privilegiado
- Es híbrido pq le podemos sumar o sacar módulos
- Libre uso, estudio, distribución y mejora y publicación

1991 primer versión 0.0.2

Recompilo el kernel para:

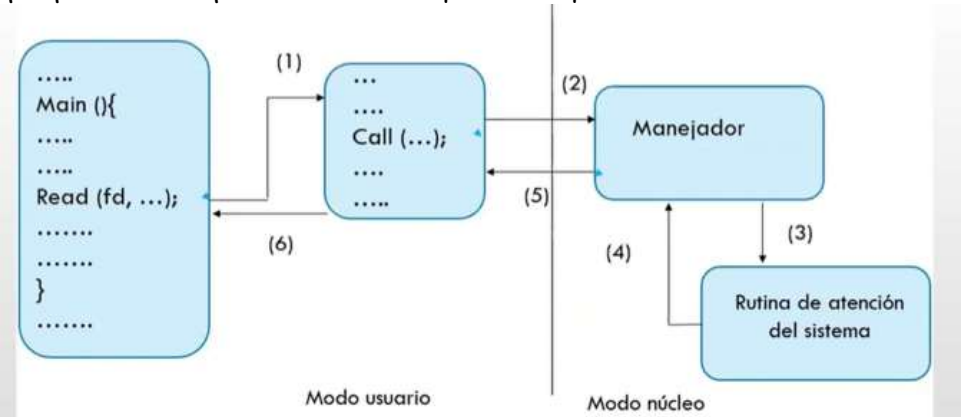
- Soporte de nuvos dispositivos
- Nuevas funcionalidades
- OPtimizar cositas
- Adaptar al sistema en donde está corriendo
- Corregir bugs

No lo hace nadie, podrías hacerlo pero x .
NEcesitas el compilador de C gcc, make para ejecutar directivas

SYSTEM CALLS:
Mecanismo que un proceso de usuario usa oara solicitar u servicio al so
Procesos de usuario ejecutan cosas en un entorno citrikadi suyo. Tienen el espacio de direcciones limtado.
Si necesita algo de fuera de ahí tiene que pedirle al so.

Puede ser que el programa invoque una system call o que llame una api para pedir los recursos.
Usar una librería me permite mudarlo de so y demás, pq yo hago la operación y la librería es la que se ocupa de eso.
EJ: en c agregamos librerias para trabajar y operar de manera sencilla

El so es como un servidor que recibe requerimientos y determina el mejor mecanismo
Tiene mecanismos y parámetros para devolver valores.
LIBC es en unix la api que define eso
Ejemplo: cuadno hago un read, lo que hago es llamar a la funcion read que está en la libreria y recibe eñ archivo al que quiero leer, en qué area de memoria y cuántos bytes.



La api es como un wrapper intermediario que llama para que se haga efectivamente el call

Agarro en la pila y pongo los parámetros que hay que pasarle a la fn para que se pueda llamar. Está implementada en la biblioteca del lenguaje de programacion. Ahí ejecuto el código especifico de la lib. (TODAVÏA EN MODO USER)

pero medio bajo nivel

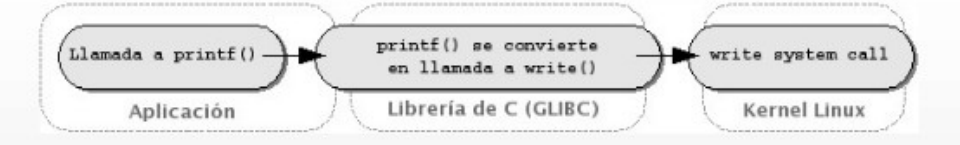
PASOS

Una vez que termina se devuelve el control a la librería en modo usuario, se sacan las cosas de la pila y se restaura

CUando hago el read lo hago en modo usuario, la librería es quien hace la syscall. Se podría hacer de una pero tendría que sa ver cómo están implementadas un montón de cosas de la arquitectura

Se hace cambio de contexto? Ni Hay cambio de modo pero como el código está dentro de cada proceso, puede tomar cosas de la pila? No entendí

Sys call write.



Codigo de la librería llama a la syscall que necesito. Tienen un nro unívoco. Define también cuáles son los parámetros que necesita.

Dispatcher usa el nro de la syscall para saber qué parámetros necesita la fnción e ir a buscarlos a la pila

Ejecuto código de mi progrma. Hago print f. cuando lelga a la librería se convierte en write. Lo que hace la lib es escribir en un registro el nro de la syscall que quiero ejecutar. Como lleva parámetros, los pone en los parámetros necesarios. Al salir pone un 0 para indicar que salio bien el print.
Si invocas la exepción sin la librería, puedo tener un problema pq yo paso parámetros. Puede ser que cuando tenga que retornar algo (0 si no hubo error, 1 si sí, etc) y si pongo 27? Explota. Puedo tener kernel panic pq se ejecuta en modo kernel lo de la syscall
La librería, el wrapper, va a chequear que no hagas cagada. Pero si lo haces desde tu programita capa la cagas

☑Categorías de system calls:

✓Control de Procesos

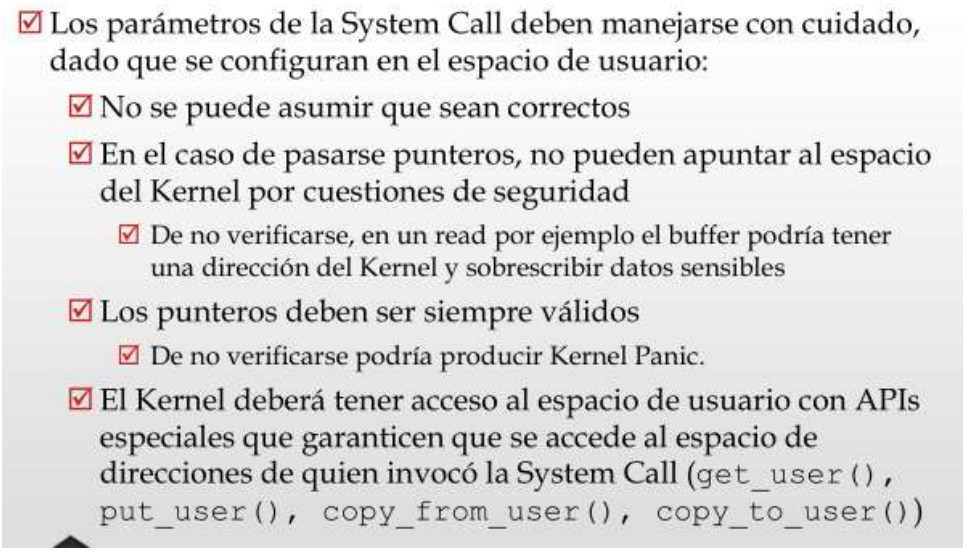
✓Manejo de archivos

✓Manejo de dispositivos

✓Mantenimiento de información del sistema

✓Comunicaciones

En linux no se puede pasar más de 6 parámetros.
Lo primero que hace el dispatcher cuando se hace una interrupción es ver el número en la tabla y ejecuar las funciones que toquen



Get uer es por ejemplo para leer de el espacio de mem del usuario. Put user para escribir. Da seguridad para que no escribas en el espacio del kernel