Clase 1- Kernel

miércoles, 12 de marzo de 2025 18:09

Sistema operativo: cualquier cosa que le da vida a un hardware. Intermediario entre el usuario y hardware
Necesita procesador y memoria para usarse

- Gestiona hw
- Controla ejecución de procesos
- Es interfaz entre apps y hw

Oculta el hw, presenta a los programas abstracciones más fáciles de manejar

Administracioón de recursos:

- Usuario no sabe que se ejeuta y cuando y como
- Da servicios. Cuando un programa de usuario quiere hacer algo le pide al so con system call
- Multiplexa en tiempo y en espacio. SI tengo una cpu, solo puedo ejecutar un proceso en simultáneo. SO también usa cpu. La onda es que los procesos operen como si tuvieran la cpu para ellos solos.
- También hay programas que son de muchas gb y no tenemos tanta ram. Entonces la idea es ue en espacio multiplexee memoria. Divide lo que tiene y le da un cachito a cada uno. Una parte está en simultáneo en memoria, se va cargando

Cositas que nos da el so

- Comodidad: es más fácil usar el hw
- Eficiencia: usa mejor los recursos. Tiene que sacarle provecho al hw
- Evolución: tiene que permitir la introducción de funciones nuevas sin interferir con lo que ya hay, sin arrancar de 0 todo

Debate: que es el so? Solo el kernel? Si pero sin la shell no puedo interactuar, pierdo comodidad. Definición estricta diria que es solo el kernel

Qué es el kernel?
Porción de codigo. Algunas funciones en memoria y después voy a buscar lo que voy necesitando. Administra los recursos

- Maneja memoria
- Maneja cpu
- Administra procesos
- Conmunicación
- Administración de memoria y de cpu
- Facilita acceso seguro al hardware
- Usa system calls: los procesos no pueden hacer cosas locas, pero le pueden pedir permiso al so y ahi ven.
- Los procesos de usuario son clientes del so. No pueden hacer lo que quieran
- Kernel atiende a los procesos siguiendo algun criterio (equidad o lo que sea)

Kernel se ejecuta en modo supervisor/privilegiado.

- Acceso a todo

Procesos se ejecutan en modo usario
Cuando un proceso se ejecuta se le da un espacio de direcciones de memoria. Si intenta acceder a algo fuera de eso no se permite, pq podria acceder a cualquier lado o moidificar cosas que no le corresponden

Como hace el cpu para darse cuenta de que un proceso está intentando ejecutar algo fuera de su espacio? (si para checkear eso necesitaria cpu) con ayuda del hardware

Lo último que hace el so cuando alguien quiere usar la cpu es programar el hw. Setea ciertos registros (ej mmu memory management unit le dice bueno este proceso solo puede crear direcciones entre x e y, si llega a meterse en otro lado avisame)tb lo del quantum. Cuando se termina, hw despierta. Seria muy ineficiente que tenga que chequear siempre el software. Hardware hace interrupciones si pasa algo. Lo despierta

Modos de ejecución: Bit que se setea para indicar si es modo kernel o modo común
Cuando tengo un kernel muy evolucionable hay que permitir instalar drivers. Es software de 3ros. Se tienen

que ejecutar en modo kernel. Puede ser peligroso.

Cuando arranca el sistema arranca en modo administrador. Cuando se va ejecutar algo en usuario, el SO lo pone en modo user.
La interrupción es la única forma de pasar a modo Kernel.
No es el proceso de user quien lo hace

Distintos tipos de kernel según arquitectura de desarrollo.
Monolítico:

- Liniux, unix, etc
- Incluye todos los servicios del so en un solo módulo. Todo ese módulo se ejeuta en modo administrador.
- Peligroso pq un driver puede hacer de todo
- Subsistemas pueden interactuar entre si. Peligroso y difícil de mantener y administrar
- Es un programa grande que hace de todo, tiene acceso a todo.
- Muy eficiente porque no me tengo que comunicar con nadie para hacer cosas o para cambiar.
- Algunos tienen un enfoque modular. Puedo ir anexando módulos.

Monolítico: kernel chiquito, customizable

Microkernel:

- Una parte de sistema operativo se ejecuta en modo user y la otra en modo kernel.
- Mete funciones principales en kernel
- Resto de cosas son periféricas se ejecutan en modo usuario
- Menos codigo en modo admin
- +estable y seguro
- Menos eficiente, tengo que cambiar de modo e ir haciendo context switch

Hibrido:

- Tomar lo mejor de ambos mundos
- Windows, macos
- Ej lo de red lo ejecuto en modo kernel (pq quiero que se ejecute rápido y demás)
- Ejecuta ciertos servicios que en el monolitico habian sacado en modo kernel
- Punto medio

Creo que no entiend la diferencia entre micro e hibrido

Exokernel

- Evita alto nivel de abstracción
- Muy específico. Para alguuuunos dispositivos de hardware
- Da servicios muy básicos, muy minimalista
- Da libertad
- Los controladores se ejecutan en espacio de usuario
- Que haya poco código hace que sea menos propenso a vectores de ataque
- No evoluciona

De tiempo real:

- Garantiza que las operaciones se completen siempre en un tiempo espeifico.
- Administran hardware de misión crítica
- No tener que sacar a otro proceso para poder explotar una bomba
- Usa algoritmos de scheduling específicos para poder complir con los plazos
- Carga todo el proceso de una en memoria. No tener que atender un fallo de página que me hace perder el tiempo
- Soporta multtasking apropiativo para que las tareas de alta prioridad suspendan a las de menos