Apunte de Sistemas Operativos FIUBA

lcondoriz

May 2023

Índice

1. Introducción **3**

2. El kernel **4**

3. Introducción: x86 **9**

4. El Proceso **9**

5. Scheduling o Planificación de Procesos **9**

6. La Memoria **9**

7. Paciales **9**

1. Introducción

Resumen hecho por lcondoriz.

¿Qué es un sistema operativo?

En un sistema operativo los usuarios interactúan con aplicaciones, estas aplicaciones se ejecutan en un ambiente que es proporcionado por el sistema operativo. A su vez el sistema operativo hace de mediador para tener acceso al hardware del equipo. La principal forma para lograr esto es mediante el concepto de **virtualización**. Esto significa que el sistema operativo toma un recurso físico (*Ej. Memorias, procesadores, persistencia*) y lo transforma en algo mas general y fácil de usar.

El sistema operativo unix esta conformado por:

- **Kernel:** Es el programa central del sistema operativo que tiene control total sobre todo en el sistema. Facilita las interacciones entre el hardware y el software y gestiona los recursos como la memoria, el procesador, los dispositivos y las interrupciones. Es el pedazo de código que esta interactuando con privilegios absolutos sobre el harware y provee una interfaz a los usuarios
- **Drivers:** Son programas que permiten al kernel comunicarse con los dispositivos de hardware y controlar su funcionamiento
- **Syscall:** Es la interfaz que permite a los programas de usuario solicitar servicios al kernel mediante llamadas al sistema
- **System utilities:** Son programas que realizan funciones básicas del sistema operativo como la administración de archivos, procesos, usuarios, redes, etc. Ejemplos ls, cat, mv, pwd...etc.
- **Disk pkgs:** Son paquetes de software que se instalan en el disco duro y que proporcionan funcionalidades adicionales al sistema operativo3.
- **Entorno gráfico:** Es la parte del sistema operativo que permite al usuario interactuar con el sistema mediante una interfaz visual basada en ventanas, iconos, menús, etc.
- **Usuario:** Es la persona que utiliza el sistema operativo y sus aplicaciones.

Modos de ejecución de un SO:

- **Kernel mode:** Es el modo privilegiado en el que se ejecuta el sistema operativo y tiene acceso completo y sin restricciones al hardware y a toda la memoria. Gracias a la existencia del kernel, los programas son independientes del hardware subyacente.

- **User mode:** Es el modo restringido en el que se ejecutan las aplicaciones y tiene un espacio de direcciones virtuales privado y limitado. El modo usuario no puede acceder directamente al hardware ni a las direcciones de memoria reservadas para el sistema operativo. El modo usuario debe hacer llamadas al sistema para solicitar servicios al sistema operativo. El modo usuario es también llamado modo esclavo, estado problemático o modo restringido.

2. El kernel

En un sistema operativo, el kernel (núcleo en español) es la parte central que actúa como intermediario entre el hardware y el software. Es el componente esencial del sistema operativo y se encarga de administrar los recursos del sistema, proporcionando servicios básicos a las aplicaciones y controlando el acceso y la comunicación con el hardware.

El kernel tiene varias responsabilidades clave, entre las que se incluyen:

1. Gestión de memoria: El kernel se encarga de asignar y liberar memoria para los procesos y programas en ejecución, asegurándose de que cada proceso tenga acceso a la cantidad de memoria necesaria y evitando conflictos.

2. Gestión de procesos: El kernel administra los procesos en ejecución, asignándoles los recursos necesarios, programando su ejecución y asegurándose de que se ejecuten de manera segura y eficiente. También maneja la planificación de la CPU, decidiendo qué procesos se ejecutan y durante cuánto tiempo.

3. Gestión de dispositivos: El kernel proporciona una capa de abstracción para los dispositivos de hardware, permitiendo que las aplicaciones accedan a ellos de manera uniforme. Controla la comunicación y el acceso a los dispositivos, gestionando los controladores correspondientes.

4. Gestión de archivos: El kernel proporciona servicios para crear, leer, escribir y eliminar archivos en el sistema de almacenamiento. También maneja la organización y el acceso a los directorios y archivos del sistema.

5. Seguridad y control de acceso: El kernel controla el acceso a los recursos del sistema, aplicando políticas de seguridad y garantizando que las aplicaciones y los usuarios solo puedan acceder a los recursos permitidos.

El kernel es entonce la capa de software de más bajo nivel en la computadora.[apunte]

Ejecución Directa

La ejecución directa de un programa es un concepto simple, significa, correr el programa directamente en la CPU. Esto otorga la ventaja de tener rapidez. Aunque también esto viene con algunos problemas. Estos son: ¿Como se asegura el OS que el programa no va a hacer nada que el usuario no quiere que sea hecho? ¿Como hace el OS para pausar la ejecución de ese programa y hacer que se ejecute otro?.

Debido a esto se necesita limitar la ejecución directa. Hoy en día esto ya no existe en un sistema operativo serio.

Limitar la Ejecución Directa

Para poder limitar la ejecución directa se necesitan ciertos mecanismos de hardware:

- Dual Mode Operation - Modo de operación dual.
- Privileged Instructions - Instrucciones Privilegiadas.
- Memory Protection - Proteccion de Memoria.
- Timer Interrupts - Interrupciones por temporizador.

Modo Dual de Operaciones

El modo de operación dual, es un mecanismo que proveen todas los procesadoresa y algunos microprocesadores modernos. El hardware debe tener la capacidad de funcionar en dos modos diferentes: modo kernel (o modo supervisor) y modo usuario (o modo usuario final). El modo kernel tiene privilegios más altos y acceso completo a todos los recursos del sistema, mientras que el modo usuario tiene acceso limitado y restringido a ciertos recursos. El sistema operativo se ejecuta en modo kernel, mientras que las aplicaciones de usuario se ejecutan en modo usuario.

Existen dos modos operacionales utilizados de la CPU :

- **Modo Usuario o User Mode:** que ejecuta instrucciones en nombre del usuario.
- **Modo Supervisor o Kernel o Monitor:** que ejecuta instrucciones en nombre del kernel del sistema operativo y estas son instrucciones privilegiadas.

Protección del Sistema:

¿Cual es el hardware necesario para que el kernel del sistema operativo pueda proteger a Usuarios y aplicaciones de otros usuarios?

Instrucciones Privilegiadas

Por la existencia del Modo Dual, los distintos modos poseen cada uno su propio set de instrucciones que pueden ser ejecutadas o no según el bit de modo de operación.

Protección de Memoria

El SO y los programas que están siendo ejecutados deben residir ambos en memoria al mismo tiempo (el SO debe cargar el programa, hacer que comience a ejecutarse y el programa tiene que residir en memoria para poder hacerlo), de modo que -para que la memoria sea compartida de forma segura- el SO debe poder configurar el hardware de forma tal que cada proceso pueda leer y escribir su propia porción de memoria.

Timer Interrupts (Interrupciones por temporizador)

Es un mecanismo que periódicamente le permita al kernel desalojar al proceso de usuario en ejecución y volver a tomar el control del procesador, y así de toda la máquina.

Visión General (Definiciones concretas)

Sistema Operativo: Es el software que controla los recursos de hardware de una computadora y provee un ambiente bajo el cual los programas pueden ejecutarse.Habitualmente a este software se lo llama el Kernel.

Gracias a la existencia del kernel los programas son independientes del hardware subyacente, es decir, se comunican con el kernel y no con el hardware directamente.

El kernel es entonces la capa de software de más bajo nivel en la computadora. El kernel es un programa.

Modos de Transferencia

Formas de alternar entre modo usuario y modo Kernel.

De Modo Usuario a Modo Kernel

- interrupciones: son eventos asíncronos que ocurren en el hardware o en el software que requieren la atención del kernel.
- excepciones del procesador: son eventos síncronos que ocurren en el procesador como resultado de la ejecución de una instrucción.
- ejecución de system calls (llamadas al sistema): son eventos síncronos que ocurren cuando una aplicación de usuario ejecuta una instrucción especial que le pide al kernel que realice una acción en su nombre.

Interrupciones

Una interrupción es una señal asincrónica enviada hacia el procesador de que algún evento externo ha sucedido y pueda requerir de la atención del mismo.

El procesador está continuamente chequeando si una interrupción es disparada. Cuando se dispara una interrupción, el procesador completa o detiene la instrucción que se esté ejecutando, guarda todo el contexto y comienza a ejecutar el manejador de esa interrupción en el Kernel.

El orden de prioridad de las interrupciones (según BCH) es:

- Errores de la Máquina.
- Timers.
- Discos.
- Network devices.
- Terminales.
- Interrupciones de Software.

Excepciones del Procesador

La otra forma por la cual se necesitaría pasar de modo usuario a modo kernel es por un evento de hardware causado por un programa de usuario. Por ejemplo, si un programa de usuario intenta dividir por cero, el procesador genera una excepción de división por cero. El procesador entonces pasa al modo kernel y ejecuta el manejador de la excepción de división por cero. Acceder fuera de la memoria del proceso, intentar escribir en memoria de solo lectura, ntentar ejecutar una instrucción privilegiada en modo usuario, etc.

System Calls

Las System Calls son funciones que permiten a los procesos de usuario pedirle al kernel que realice operaciones en su nombre. Las system calls son la interfaz entre el kernel y las aplicaciones de usuario.

System Calls

Una system call (llamada al sistema) es un punto de entrada controlado al kernel permitiendo a un proceso solicitar que el kernel realice alguna operación en su nombre [KER](cap. 3).

Algunas características generales de las system calls son:

- Una system call cambia el modo del procesador de user mode a kernel mode, por ende la CPU podrá acceder al área protegida del kernel.

- El conjunto de system calls es fijo. Cada system call esta identificada por un único número, que por supuesto no es visible al programa, este sólo conoce su nombre.
- Cada system call debe tener un conjunto de parámetros que especifican información que debe ser transferida desde el user space al kernel space.

Llamada a una System Call

Desde el punto de vista de un programa llamar a una system call es mas o menos como invocar a una función de C. Por supuesto, detrás de bambalinas muchas cosas suceden:

1. El programa realiza un llamado a una system call mediante la invocación de una función wrapper (envoltorio) en la biblioteca de C.
2. Dicha función wrapper tiene que proporcionar todos los argumentos al system call trap_handling. Estos argumentos son pasados al wraper por el stack, pero el kernel los espera en determinados registros. La función wraper copia estos valores a los registros.
3. Dado que todas las system calls son accedidas de la misma forma, el kernel tiene que saber identificarlas de alguna forma. Para poder hacer esto, la función wrapper copia el número de la system call a un determinado registro de la CPU.
4. La función wrapper ejecuta una instrucción de código maquina llamada trap machine instruction (int 0x80), esta causa que el procesador pase de user mode a kernel mode y ejecute el código apuntado por la dirección 0x80 (128) del vector de traps del sistema.
5. En respuesta al trap de la posición 128, el kernel invoca su propia función llamada syste_call() (arch/i386/entry.s) para manejar esa trap. Este manejador:
   a) graba el valor de los registros en el stack del kernel.
   b) verifica la validez del numero de system call.
   c) invoca el servicio correspondiente a la system call llamada a través del vector de system calls, el servico realiza su tarea y finalmete le devuelve un resultado de estado a la rutina system_call().
   d) se restauran los registros almacenado en el stack del kernel y se agrega el valor de retorno en el stack.
   e) se devuelve el control al wraper y simultáneamente se pasa a user mode.
6. Si el valor de retorno del la rutina de servicio de la system call da error, la función wrapper setea el valor en errno.

Tipos de Kernel

Existen básicamente dos tipos de estructuras de kernel:

- **Monolítico:** El kernel es un único programa (en realidad proceso) que se ejecuta continuamente en la memoria de la computadora intercambiándose con la ejecución de los procesos de usuario. Contiene todos los servicios del sistema operativo en el mismo espacio de direcciones.
- **Microkernel:** Es un kernel que se ejecuta en modo kernel y contiene solo los servicios esenciales del sistema operativo, como la gestión de memoria, la planificación de procesos y la comunicación entre procesos. Todos los demás servicios del sistema operativo se ejecutan como procesos de usuario fuera del kernel. El kernel proporciona una interfaz de programación de aplicaciones (API) para que los procesos de usuario puedan comunicarse con los servicios del kernel. El kernel microkernel es el tipo de kernel utilizado por los sistemas operativos macOS e iOS de Apple.
- **Híbrido:** Es un kernel que se ejecuta en modo kernel y contiene algunos servicios del sistema operativo en el mismo espacio de direcciones, mientras que otros servicios se ejecutan como procesos de usuario fuera del kernel. El kernel híbrido es el tipo de kernel utilizado por el sistema operativo Windows de Microsoft.

Existen básicamente dos tipos de estructuras de kernel:

3. Introducción: x86

Ley de Moore

En 1965 Gordon Moore, cofundador de Intel formuló una ley empírica que se ha podido constatar hasta nuestros días que dice:

"Aproximadamente cada dos años se duplica el número de transistores en un microprocesador por unidad de área"

Arquitectura x86: Hardware

4. El Proceso

5. Scheduling o Planificación de Procesos

6. La Memoria

7. Paciales

¿Que es el stack ? Explique el mecanismo de funcionamiento del stack para x86 de la siguiente funcion int read(void *buff, size_t num,

int fd);.Como se pasan los parametros, direccion de retorno?.

El stack o pila es una estructura de datos que almacena información de forma temporal y ordenada, siguiendo el principio LIFO (Last In, First Out), es decir, el último en entrar es el primero en salir. El stack se usa para guardar los datos locales de una función, las direcciones de retorno de las llamadas a funciones y los parámetros que se pasan a las funciones.

Para la función read(void *buff, size_t num, int fd), que lee num bytes del archivo identificado por fd y los almacena en el buffer buff, se puede usar el stack para pasar los parámetros de la siguiente manera:

- Se empujan los parámetros al stack en orden inverso, es decir, primero fd, luego num y finalmente buff.
- Se llama a la función read con la instrucción call, que empuja la dirección de retorno al stack y salta a la etiqueta de la función.
- Dentro de la función read, se accede a los parámetros usando el registro ebp (base pointer) como referencia. El registro ebp se usa para guardar el valor del registro esp (stack pointer) al entrar en la función, y así poder acceder a los parámetros y variables locales sin importar cómo cambie el esp durante la ejecución de la función
- Se usa la convención cdecl para limpiar el stack después de la llamada a la función. Esta convención establece que el código que llama a la función es responsable de restaurar el esp al valor que tenía antes de empujar los parámetros. Esto se hace sumando al esp el tamaño total de los parámetros.

Un posible código en ensamblador x86 para este ejemplo sería:
; Código que llama a la función read ; Supongamos que fd = 3 (stdin), num = 100 y buff apunta a una zona de memoria reservada mov eax, 3 ; fd push eax ; empujar fd al stack mov eax, 100 ; num push eax ; empujar num al stack mov eax, buff ; buff push eax ; empujar buff al stack call read ; llamar a la función read add esp, 12 ; limpiar el stack (3 parámetros de 4 bytes cada uno)
; Código de la función read read: push ebp ; guardar el valor anterior de ebp mov ebp, esp ; copiar el valor de esp a ebp ; Ahora los parámetros se pueden acceder como [ebp+8], [ebp+12] y [ebp+16] ; Aquí iría el código para leer del archivo y escribir en el buffer ; usando las instrucciones syscall o int 80h mov esp, ebp ; restaurar el valor de esp pop ebp ; restaurar el valor de ebp ret ; retornar a la dirección guardada en el stack

Referencias

[1] Arboles de Decisión. En: (). URL: https://www.ibm.com/es-es/topics/decision-trees.

[2] Colab Regresion Logistica. En: (). URL: https://colab.research.google.com/drive/1JbRUFa5hnijNQdJ_HLywhMJrICkColMJ?authuser=1#scrollTo=sIl68yHmh0yS.

[3] Video de YouTube KNN. En: 31 min (). URL: https://www.youtube.com/watch?v=cH-kUai4Boo.