INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – Tema 1, parte 2

Funciones principales de un sistema operativo:

- Dar abstracciones de alto nivel a procesos de usuario.
- Administrar eficientemente la CPU.
- Administrar eficientemente la memoria.
- Brindar asistencia al proceso de E/S por parte de los procesos.
- El sistema operativo necesita de CPU y memoria, el clock molesta al SO para que se ponga a cargo.
- Hardware no distingue interrupciones.

¿Qué problemas debe evitar un SO?

- Que un proceso se apropie el CPU.
- Que un proceso intente ejecutar instrucciones de E/S o privilegiadas →por ej: un programa feo que pide escribir en el bloque 500 del disco, ta mal..
- Que un proceso intente acceder a una posición de memoria fuera de su espacio permitido. → se deben proteger los espacios de direcciones.

El SO debe:

- Gestionar el uso de cpu
- Detectar intentos de ejecución de instrucciones de E/S ilegales.
- Detectar accesos ilegales a memoria.
- Proteger el vector de interrupciones → Rutinas de atención de interrupciones: recordemos arquitectura, una interrupción es un bloque de código que se ejecuta cuando se interrumpe el secuenciamiento del procesador durante la ejecución de un proceso. Se tratan de dispositivos de E/S, estas interrupciones son de ayuda cuando ocurren ciertos eventos.

Apoyo del Hardware ante los problemas que pueden surgir:

- Modos de ejecución: esto define limitaciones en el conjunto de instrucciones que se puede ejecutar en cada modo.

    - Hay un bit en la CPU que indica el modo actual en el cuál se ejecuta la CPU. Este bit de modo se almacena en un registro llamado registro de palabra de estado del programa (PSW) . Si

alguien quiere saber en qué modo está ese sistema, puede verlo en el registro de PSW.

- ¿A qué se refiere modo? → a lo que la CPU puede hacer en función al modo que está.
- Las instrucciones privilegiadas deben ejecutarse en modo **Supervisor o Kernel** (se pueden ejecutar todas las instrucciones y está todo permitido)→ Se necesitan acceder a estructuras del kernel, o ejecutar código que NO es del proceso, código del kernel.
- En modo **Usuario**, el proceso puede acceder sólo a su espacio de direcciones, es decir, a direcciones propias, es un modo acotado a lo que se puede hacer, no todas las instrucciones se pueden hacer (las instrucciones privilegiadas no se permiten acá).

- El kernel del SO se ejecuta en modo supervisor.

- El resto del SO y los programas de usuario se ejecutan en modo usuario (teniendo un subconjunto de instrucciones permitidas).

- **TENER EN CUENTA**:

  - Cuando se arranca el sistema arranca con el bit en modo SUPERVISOR.

  - Cada vez que se comienza a ejecutar un proceso de usuario, el bit se debe poner en modo usuario. → mediante instrucción especial.

  - Para que el bit vuelva al modo Kernel, debe ocurrir cualquier interrupción que provoca que el bit vuelva al modo Kernel.
    - Se trata de la única forma de pasar al modo kernel.
    - No es el proceso de usuario quien hace el cambio explícitamente.
    - El SO determina cuando se debe volver al modo usuario → bit de nuevo en modo usuario.

- ¿Cómo actúa la interrupción? → cuando el proceso de usuario intenta por sí mismo ejecutar instrucciones que pueden causar problemas (intento de ejecutar instrucción privilegiada). El HW lo detecta como operación ilegal y produce una interrupción al SO. El Kernel es el responsable de atender las interrupciones.

- El mecanismo de los modos de ejecución + interrupciones nos ayuda a que no ocurran errores de ejecución de instrucciones y por tanto proteger la CPU.

- Cuando un programa se bloquea en modo usuario, a lo sumo se escribe un suceso en el registro de sucesos. Si el bloqueo se produce estando en modo supervisor se genera la BSOD (pantalla azul de la muerte).

- Un poco de historia….

El procesador Intel 8088 no tenía modo dual de operación ni protección por hardware.

En MsDos (SO) las aplicaciones pueden acceder directamente a las funciones básicas de E/S para escribir directamente en pantalla o en disco (Todo en modo Kernel).

- **Resumiendo**:
  - Modo kernel (libre):
    - Gestión de procesos: creación y finalización, planificación, intercambio, sincronización, soporte para comunicación entre procesos.
    - Gestión de memoria: reserva de espacio de direcciones para procesos, swapping, gestión, y paginas de segmentos.
    - Gestión de E/S: gestión de buffers, reserva de canales de E/S y de dispositivos de los procesos.
    - Funciones de soporte: gestión de interrupciones, auditoria, monitoreo.

  - Modo usuario (restrictivo):
    - Debugging de procesos, definición de protocolos de comunicación, gestión de apps (compiladores, editores, aplicaciones de usuario).

- Se llevan a cabo tareas que no requieren accesos de privilegio.
- No se puede interactuar con hardware.
- El proceso trabaja en su propio espacio de memoria.

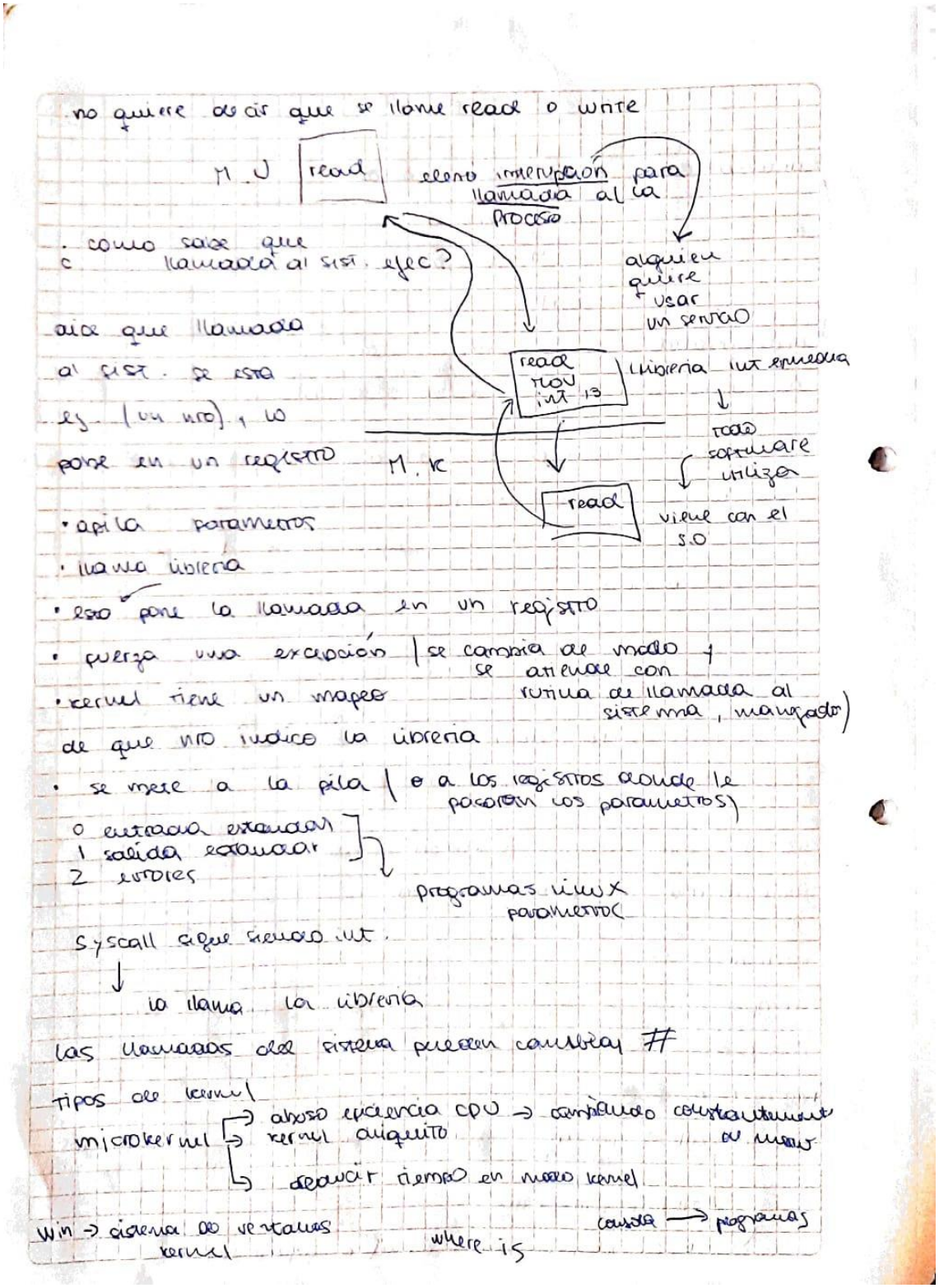
- Interrupción de clock: se debe evitar que un proceso se apropie de la CPU.
  - La interrupción por clock ocurre siempre, ayuda a evitar que un proceso se apropie del CPU, sopapea al sistema operativo para que siempre controle.
  - Se implementa normalmente a través de un clock y un contador.
  - El kernel le da valor al contador que se decrementa con cada tick de reloj y al llegar a cero puede expulsar al proceso para ejecutar otro.
    - Las instrucciones que modifican el funcionamiento del reloj son privilegiadas.
    - Se le asigna al contador el valor que se quiere que se ejecute un proceso (cantidad de tiempo).
    - Se la usa también para el cálculo de la hora actual, basándose en cantidad de interrupciones ocurridas cada tanto tiempo y desde una fecha y hora determinada.
- Protección de la memoria: se deben definir límites de memoría a los que puede acceder cada proceso (registros base y límite).
  - Delimitar el espacio de direcciones para cada proceso.
  - Poner limites a las direcciones que puede usar un proceso (corralito para un proceso).
  - El sistema operativo sabe las direcciones asignadas para cada proceso.
    - Por ejemplo: se usa un registro base y un registro límite. (está entre ésta dirección inicial y ésta dirección límite).
    - El kernel carga estos registros por medio de instrucciones privilegiadas. La acción de asignación de registro base y registro límite sólo es realizable en modo kernel.
    - La CPU controla que el proceso que se está ejecutando sea siempre en los límites establecidos.
      - El kernel debe proteger para que los procesos de usuarios no accedan a donde no deben.

- El kernel protege el espacio de direcciones de un proceso del acceso de otros procesos.
- Cuando una instrucción quiere entrar a un lugar donde no debe → se provoca una interrupción por software.

System Calls

- Es la forma en que los programas de usuario acceden a los servicios del SO.
- Los parámetros asociados a las llamadas pueden pasarse de varias maneras: por registros, bloques, o tablas en memoria ó la pila.
- Las system calls se ejecutan en modo kernel.
- Los System Calls son rutinas, funciones, procedimientos que los procesos de usuario pueden invocar para pedirles servicios al sistema operativo (requieren parámetros).
- Cuando se ejecuta una system call, lo que se hace es llamar a una librería que responde al llamado, dicha librería la crea el que diseña el sistema operativo. La librería se encuentra en MODO usuario. La librería identifica que llamada está siendo usada, inserta un número en un determinado registro, y FUERZA una interrupción por software → provoca un cambio de MODO, pasa al modo privilegiado. Al pasar al modo kernel, se identifica el número insertado en el registro (identificador de qué system call se realizó, incluyendo donde están los parametros) → se ejecuta la llamada. Una vez que ese proceso finaliza, se vuelve al modo usuario y se vuelve al lugar donde se invocó la interrupción (sigue el flujo de instrucciones).

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – Tema 1, parte 2



Categorías de las System Calls:

- Control de procesos
- Manejo de archivos
- Manejo de dispositivos
- Mantenimiento de información del sistema

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – Tema 1, parte 2

- Comunicaciones

Tipos de Kernel (en el sentido de organización):

- Kernel Monolítico: toda funcionalidad que debe implementar el SO se ejecuta en modo kernel.
  - ¿Ventajas?
    - Modelarlo implica menos tiempo en la resolución de las cosas.
    - Es todo más rápido → más inseguro.
- Microkernel: el kernel se trata de ser lo más chico posible, se dejan en el modo de usuario diferentes componentes que se ejecutarán para dar apoyo al kernel. Ejemplo: tenemos procesos a ejecutar, cada proceso se ejecuta en X segundos y después cambia a otro proceso. La selección del siguiente proceso lo hace el modo usuario. El modo kernel para el microkernel reduce al máximo lo que sí o sí necesita el modo kernel para estar en ejecución (necesita → espacios de direcciones para asignar a procesos, comunicar procesos, planificación básica). Deja en el modo usuario parte de las cosas que pueden resolverse tranquilamente en el modo usuario.
  - ¿Ventajas?
    - Se sostiene que el modo kernel debe estar mucho menos tiempo al ser propenso a problemas (los problemas en éste modo implican graves costos para el SO).
    - Apuntan más a la seguridad del sistema.
- **La mayoría de Kernel tienen diseños monolíticos.**