**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.
* 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)

Las instrucciones privilegiadas deben ejecutarse en modo **Supervisor o 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).

**Cambio de usuario a kernel :**

* Los cambios entre modo usuario y modo kernel son fundamentales para la seguridad y estabilidad del sistema operativo. Las interrupciones son el mecanismo clave que permite este cambio de modo.
* **Diferencias entre Modo Kernel ,Modo Usuario y usuario Root:**

El modo kernel, es el nivel de privilegio más alto en un sistema operativo. Cuando el procesador está en modo kernel, tiene acceso completo a todos los recursos del sistema, incluyendo la memoria, dispositivos de hardware y todas las instrucciones de la CPU.

**Ejemplo**: El sistema operativo, especialmente el núcleo (kernel), ejecuta sus operaciones críticas en modo kernel. Esto permite que el sistema operativo controle y gestione el hardware y otros recursos del sistema.

**Acceso**: En modo kernel, los programas pueden hacer prácticamente cualquier cosa, lo que incluye manipular directamente el hardware y la memoria del sistema.

**Seguridad**: Debido a su acceso total al sistema, el código que se ejecuta en modo kernel debe ser confiable y seguro, ya que cualquier fallo podría llevar a un colapso del sistema completo.

Mientras que el modo usuario tiene restricciones para evitar fallos del sistema.

El modo usuario es un nivel de privilegio más bajo, utilizado para ejecutar aplicaciones y programas que no necesitan acceso directo al hardware o a la memoria crítica del sistema.

**Ejemplo**: La mayoría de las aplicaciones que usamos, como un navegador web o un editor de texto, se ejecutan en modo usuario.

**Acceso**: En modo usuario, los programas tienen acceso limitado a los recursos del sistema. Si necesitan realizar operaciones privilegiadas, como acceder al hardware, deben solicitar al sistema operativo que lo haga en su nombre.

**Seguridad**: Este modo está diseñado para proteger el sistema operativo de aplicaciones maliciosas o defectuosas, limitando lo que pueden hacer.

El usuario "root" en Linux es un usuario con privilegios administrativos, capaz de realizar cualquier operación en el sistema, como instalar software, modificar configuraciones críticas, etc.

**Relación con Modo Kernel:** Aunque el usuario root tiene permisos para realizar muchas acciones en el sistema, el código que ejecuta sigue funcionando en modo usuario cuando realiza tareas normales. Sin embargo, cuando el root ejecuta ciertas operaciones que interactúan directamente con el kernel (como cambiar configuraciones del sistema), el kernel las maneja en modo kernel.

**Conclusión: El modo kernel y modo usuario son conceptos de nivel de hardware que dictan cómo se ejecutan los procesos en un sistema operativo, mientras que el usuario root es una entidad de software con altos privilegios dentro del sistema operativo, pero no necesariamente ejecuta código en modo kernel.**

* **El SO tiene la responsabilidad de proteger la CPU y la memoria de accesos no autorizados. Esto es esencial para garantizar que los procesos funcionen correctamente sin interferencias.**

**Interrupciuones por reloj:**

Las interrupciones por reloj permiten al SO gestionar el tiempo de CPU entre múltiples procesos. Esto asegura que cada proceso tenga un tiempo de ejecución.

**Servicios que ofrece el SO:**

Los servicios que ofrece el SO incluyen la lectura y escritura de archivos, asi como la ejecución de programas. Estos servicios son cruciales para el funcionamiento de las aplicaciones.

Las llamadas al sistema son procedimientos que permiten a los programas solicitar servicios del sistema operativo. Estas llamadas requieren de un modo privilegiado para su ejecución.

El proceso para invocar una llamada al sistema implica un cambio de modo del usuario al modo kernel. Esto es esencial para garantizar la seguridad y el control del sistema operativo.

System Calls ( llamadas al sistema )

Las llamadas al sistema son fundamentales para que los programas interactúen con el hardware y el sistema operativo. Estas llamadas permiten la creación, sincronización y terminación de procesos de manera eficiente.

Las interrupciones son cruciales para el funcionamiento de las llamadas al sistema, y cada hardware tiene un número limitado de ellas. Esto asegura que el sistema operativo las distinga correctamente.

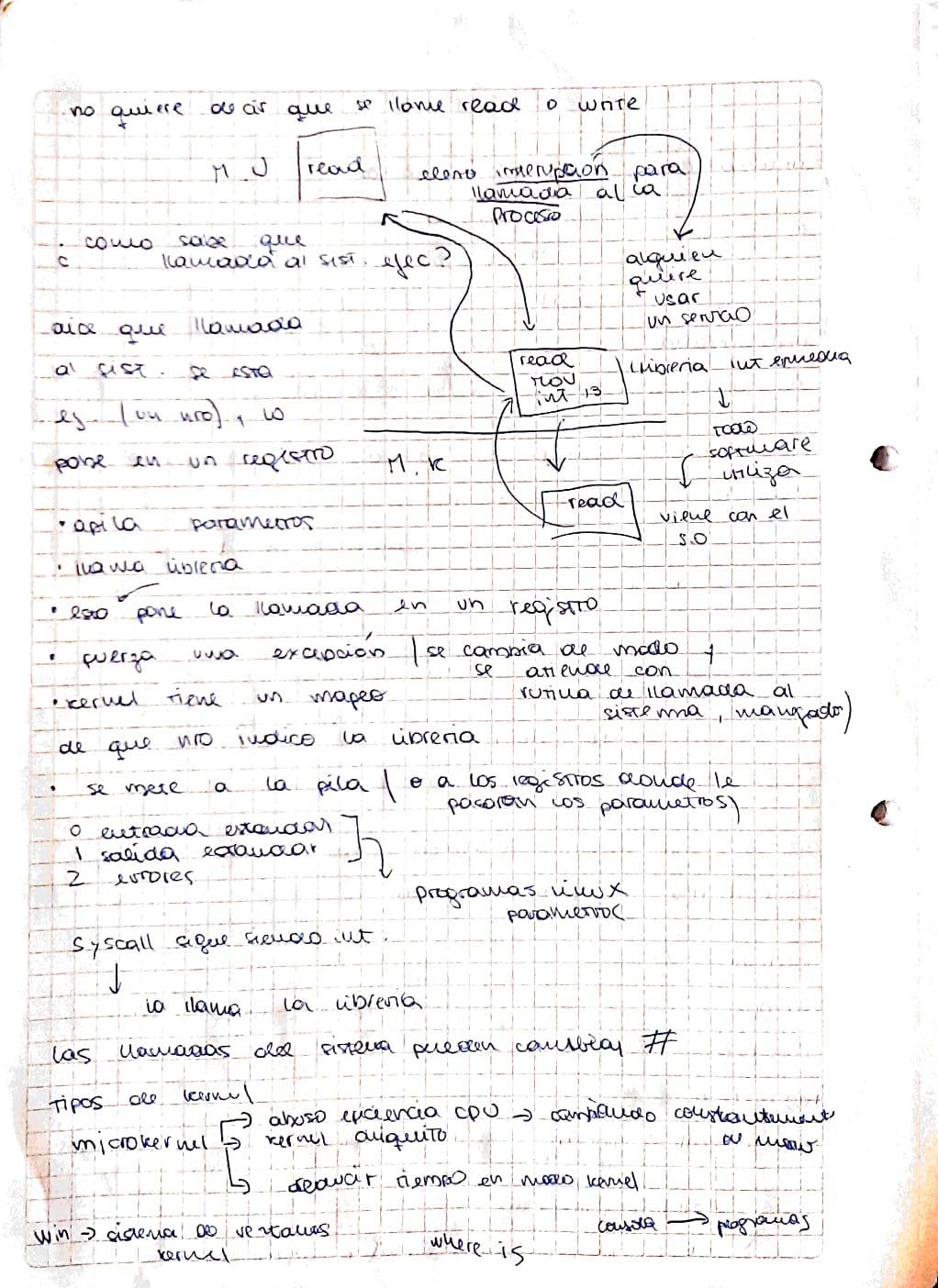
**Kernel monolítico vs microkernel:**

|  |  |
| --- | --- |
| **Kernel monolítico** | **Microkernel** |
| Implica menos tiempo en la resolución de tareas. Es decir: Cambio de modo (modo kernel), resuelvo todo lo que tenga que resolver y vuelvo al modo usuario  **Este tipo de kernel, apuntan a la performance de ejecucion** | Es necesario que este menos tiempo en modo privilegiado. Por que el modo privilegiado es el que puede causar problemas.  En modo privilegiado, si hay un error tengo mas posibilidades de tener un problema grande en el S.O  **Este tipo de kernel apuntan a la seguridad** |

* + **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.
  + Un poco de historia….
  + **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).



Categorías de las System Calls:

* Control de procesos
* Manejo de archivos
* Manejo de dispositivos
* Mantenimiento de información del sistema
* 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.**