# Problemas a evitar que se ocupa el SO

* Que un proceso se apropie de la CPU.
* Que un proceso intente ejecutar instrucciones de E/S por ejemplo.
* Que un proceso intente acceder a un área de memoria fuera de su espacio declarado

Para evitar esto el SO entre otras cosas debe

* Gestionar el uso de la CPU
* Detectar intentos de ejecución de instrucciones E/S ilegales
* Proteger el vector de interrupciones

Para hacer esto se **apoya en el HW** mediante

* Modos de ejecuciones
* Interrupción de Clock
* Protección de la memoria

# Apoyo en el HW

**Modos de ejecución**:

Modo kernel:

* Gestión de procesos: Creación y terminación, planificación, intercambio, sincronización y soporte para la comunicación entre procesos.
* Gestión de memoria: Reserva de espacio de direcciones para los procesos, Swapping, Gestión y páginas de segmentos.
* Gestión E/S: Gestión de buffers, reserva de canales de E/S y de dispositivos de los procesos.
* Funciones de soporte: Gestión de interrupciones, auditoría, monitoreo.

Modo usuario:

* Debug de procesos, definición de protocolos de comunicación, gestión de aplicaciones (compilador, editor, aplicaciones de usuario).
* En este modo se llevan a cabo todas las tareas que no requieran accesos privilegiados.
* En este modo no se puede interactuar con el hardware.
* El proceso trabaja en su propio espacio de direcciones.

Para pasar de un modo a otro se hace de maneras distintas

De kernel a usuario se hace mediante una instrucción especial.

De usuario a kernel solo se puede pasar mediante un trap o interrupción.

Tener en cuenta que el SO arranca en modo kernel.

Los traps del modo usuario se producirán porque intenta hacer una operación indebida para ese modo, por lo tanto, le pasa el mando al modo kernel, ahora si nos sale un trap estando ya en el modo kernel, ahí en Windows por ejemplo ocurriría una **pantalla azul**.

**Interrupción por Clock**:

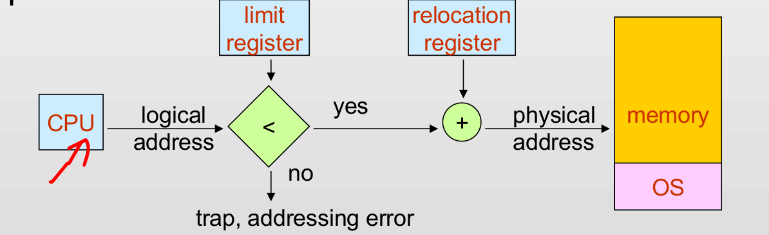
Cada tanto tiempo si el proceso no ha finalizado, directamente se finaliza el proceso para pasar a otro, de manera que un proceso no se apropie de la CPU con un while infinito, por ejemplo.

**Protección de la memoria**:

Delimitar el espacio de direcciones del proceso

Poner limites a las direcciones que puede utilizar un proceso

La CPU (El kernel) se encarga de que el proceso no intente pasar sus límites APOYANDOSE en HW



**CONTINUAR….**

# **System calls**:

Cuando un proceso necesita servicio del SO, los solicitan a través de **System Calls**, que se ejecutan en modo kernel y se invocan a través de interrupciones. (Para más detalle ver video tema 1 Clase 2 filmina 58)

# Tipos de kernel:

Monolítico: toda funcionalidad que debe implementar el SO se ejecuta en modo kernel.

Este modo implica menos tiempo en la resolución de las cosas

Se apunta a la perfomance.

Microkernel: se intenta hacer lo mas chico posible el kernel haciendo que se encargue lo que sí o sí debe ser hecho por él, implementando que el modo usuario de apoyo a este.

Este modo implica una mayor seguridad a que no haya errores que se producen en el modo kernel (lo de la pantalla azul, por ejemplo)

Se apunta a la seguridad.

La mayoría de procesadores utiliza microkernel, Linux y Windows son uno de ellos.