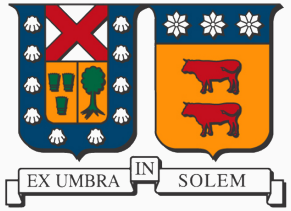


Sistemas Operativos

Organización y Diseño

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.

Índice de Contenidos

Inicio del Sistema e Interrupciones

Organización - Inicio del Sistema

- El inicio del sistema comienza con la ejecución de la BIOS.
- Ésta se encuentra en una memoria (*no volatil*) de la placa madre.
- Luego de preparar el HW carga el **boot loader**.
- El boot loader busca y carga el **kernel** del SO en memoria.
- El sistema queda operativo esperando que *suceda algo*.
- Ejemplo: GRUB permite seleccionar el Kernel a ejecutar.

Organización - Inicio del Sistema

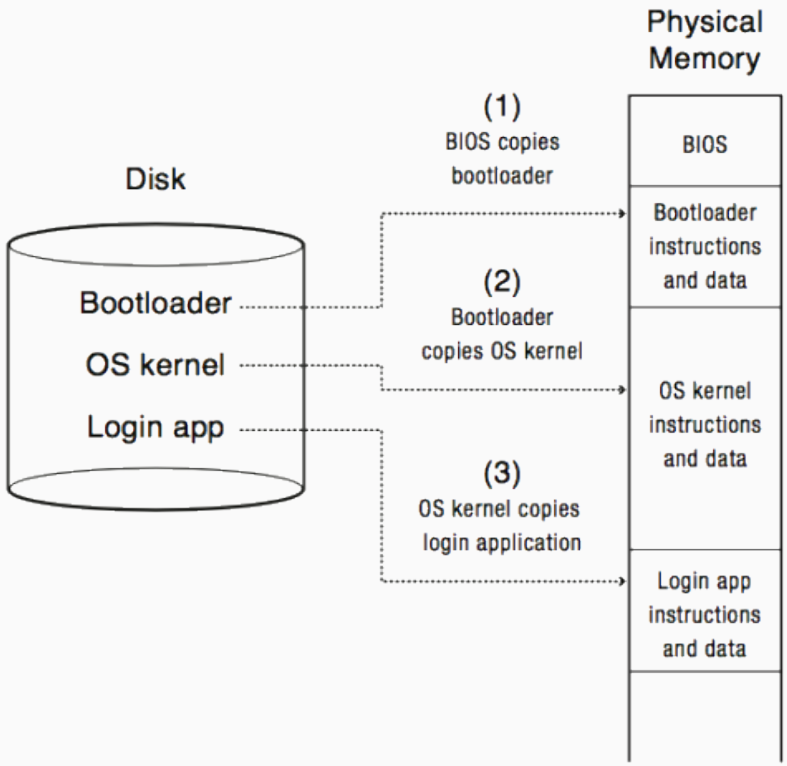


Figure 1: Inicio del Sistema

Organización - Interrupciones

- La ocurrencia de un evento se indica a través de una **interrupción**.
- Estos eventos cambian el flujo normal de ejecución de instrucciones.
- Cuando se producen, la CPU deja de hacer lo que está haciendo.
- Se revisa qué pasó, se resuelve y se continua la ejecución.
- Pueden ser producidas por hardware o software.

Definición

SO's modernos son conducidos por interrupciones (*OS interrupt driven*).

Organización - Interrupciones

- Los eventos producidos por *software* se denominan **traps**.
- Son síncronas respecto a la ejecución de instrucciones.
- Ejemplos.
  - ✓ División por cero.
  - ✓ Overflow aritmético.
  - ✓ Referencia de memoria inválida.

Organización - Interrupciones

- Los eventos producidos por *hardware* se denominan **interrupciones**.
- Son asíncronas respecto a la ejecución de instrucciones.
- Ejemplos.
  - ✓ Oprimir teclas.
  - ✓ Término de transferencia de datos.
  - ✓ Bateria baja.
  - ✓ Timer.

Timer

Dispositivo de hardware que periodicamente interrumpe al procesador.
La frecuencia con que lo hace es definida en el kernel del SO.

Organización - Interrupciones

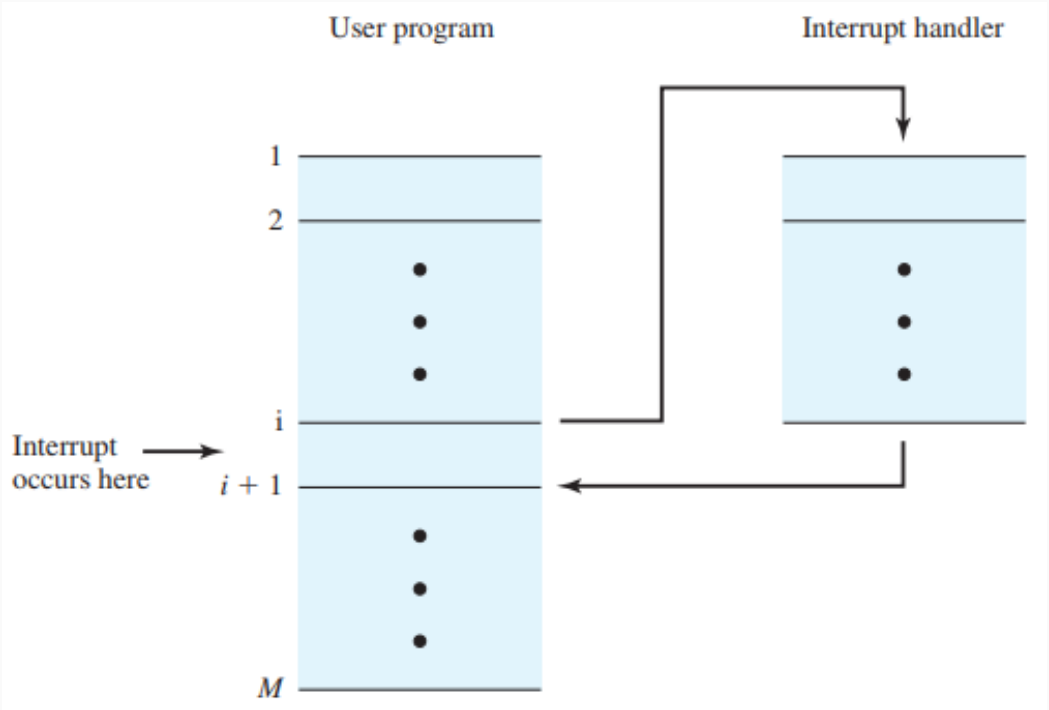


Figure 2: Evento de Interrupción

Organización - Interrupciones

- El código que resuelve un evento se denomina **interrupt handler**.
- Mientras se ejecuta:
  - ✓ se *deshabilitan temporalmente* las interrupciones.
  - ✓ no se puede bloquear. Se ejecuta hasta finalizar.
- Cuando finaliza la ejecución se vuelven a habilitar.

Enmascaramiento de Interrupciones

El SO puede deshabilitar interrupciones ante otros eventos.

Organización - Interrupciones

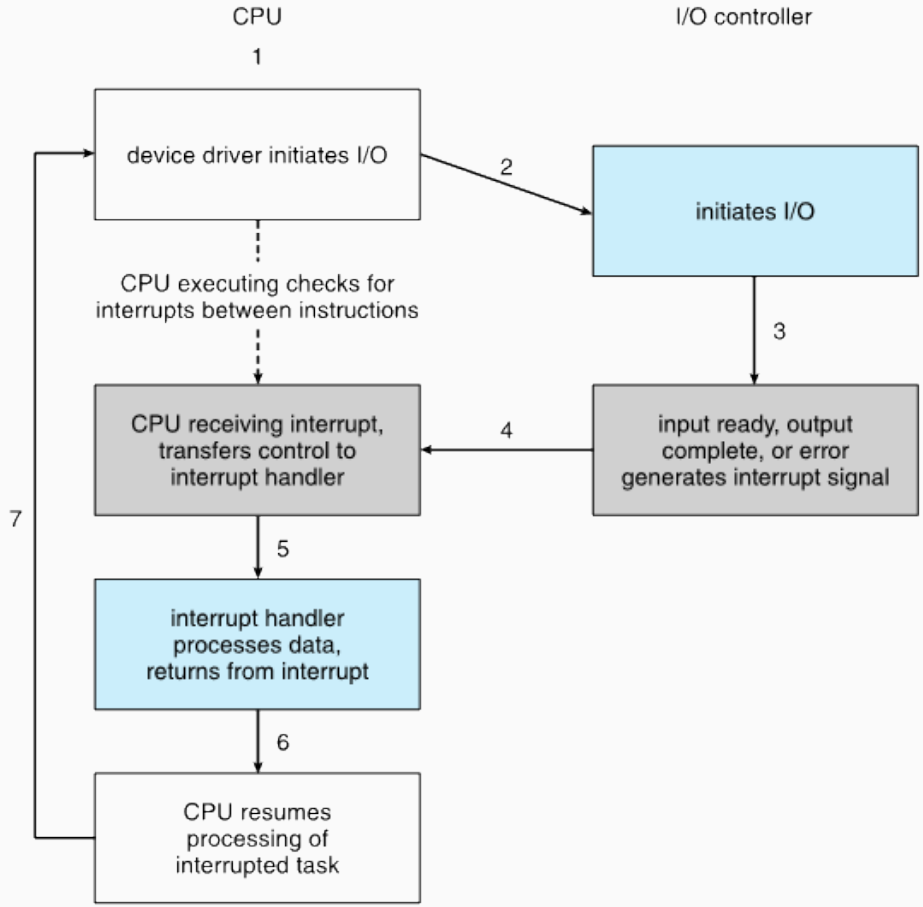


Figure 3: Evento Interrupción E/S.

Organización - Interrupciones

- El SO agrupará los gestionadores en el **vector de interrupciones**.
- Normalemente está en direcciones bajas de la memoria.
- Contiene punteros a los diferentes gestionadores.
- Otra opción es utilizar la arquitectura del **Coprocessor0**.

Organización - Interrupciones

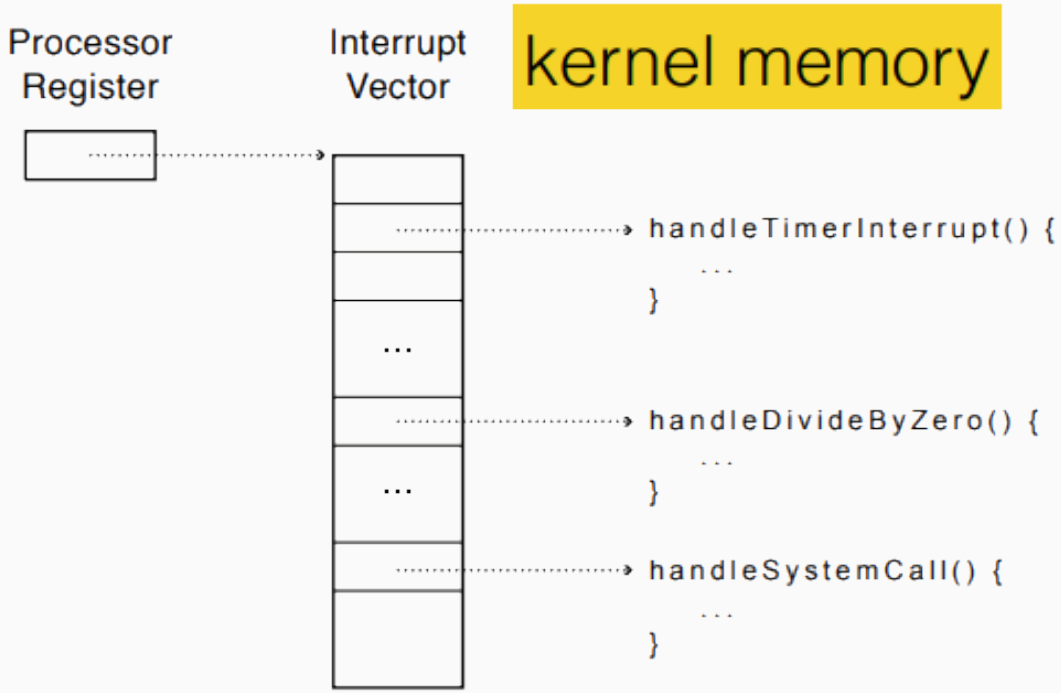


Figure 4: Vector de Interrupción

Organización - Interrupciones

- La ejecución de instrucciones por parte del procesador conlleva almacenar información sensible en ciertos registros de apoyo. i.e: dirección de la siguiente instrucción a ejecutar en el PC.
- Ante una interrupción se debe guardar esta información.
- SO's incorporan un **stack** para cada programa en ejecución.
- Estos stack's están en la memoria definida para el SO.

Organización - Interrupciones

Ejemplo: La arquitectura x86 realiza lo siguiente ante una interrupción.

- Guarda la información de los registros en el stack.
- Direccionar al vector de interrupciones.
- Ejecutar el gestionador correspondiente.
- Restaurar la información a los registros.
- Reanuda la ejecución.

Organización - Interrupciones

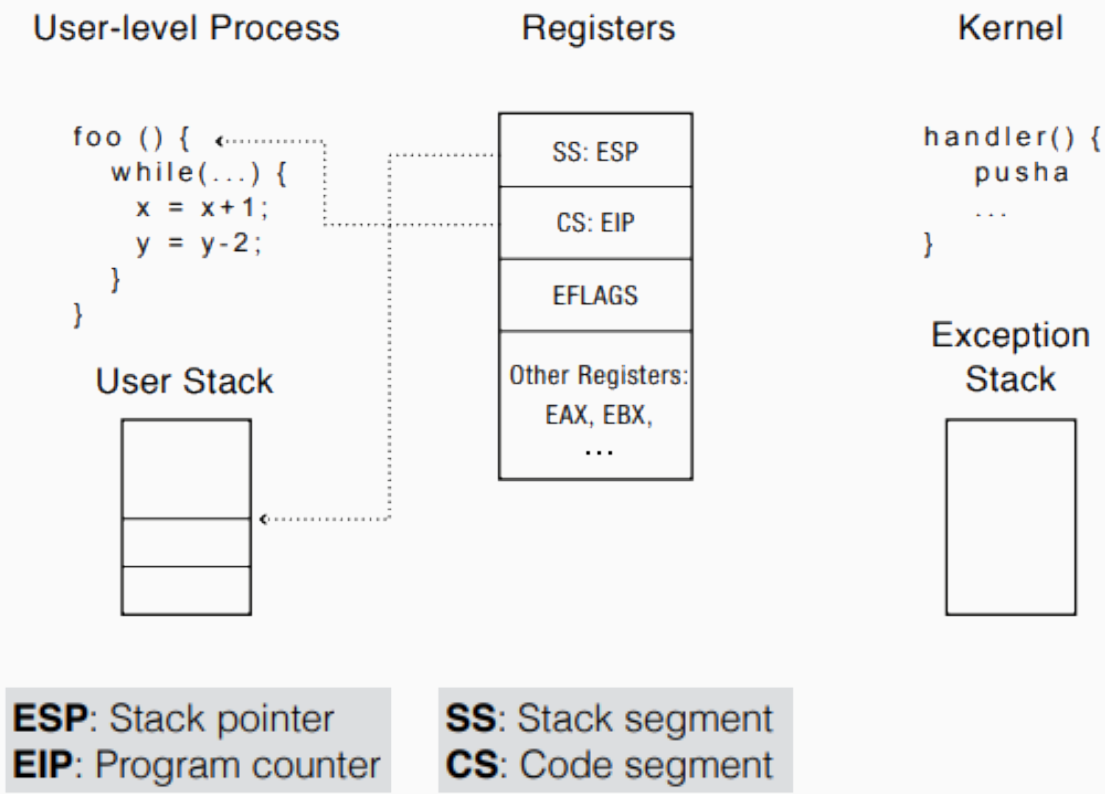


Figure 5: Antes de la interrupción

Organización - Interrupciones

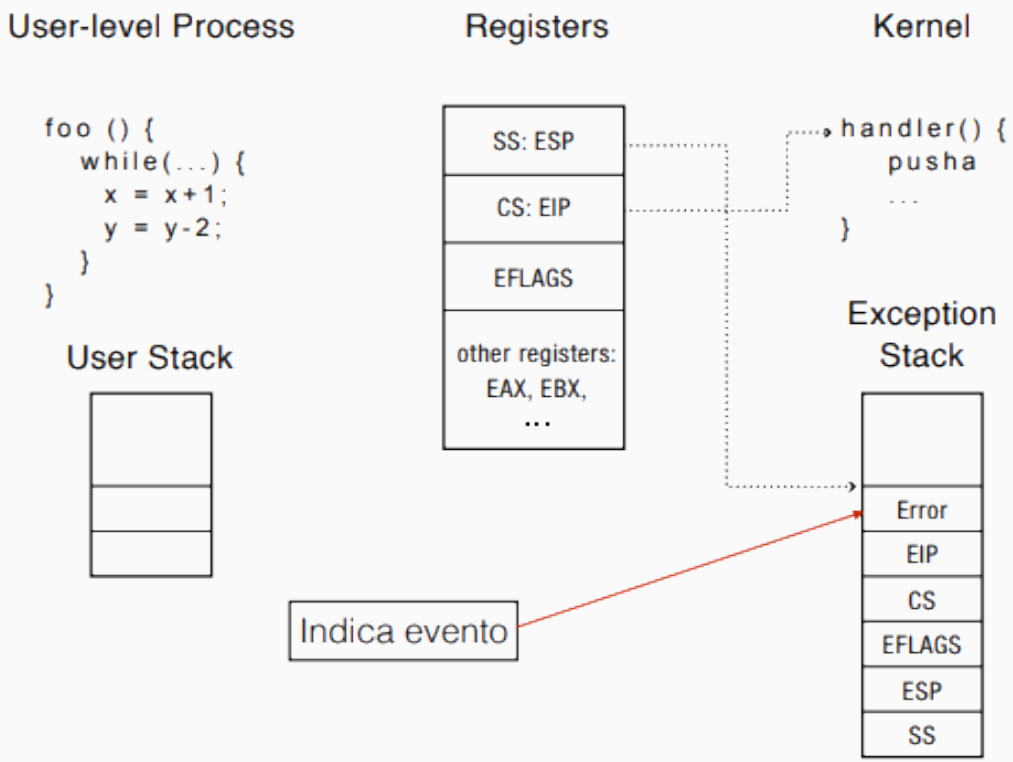


Figure 6: Justo al direccionar al gestionador

Organización - Interrupciones

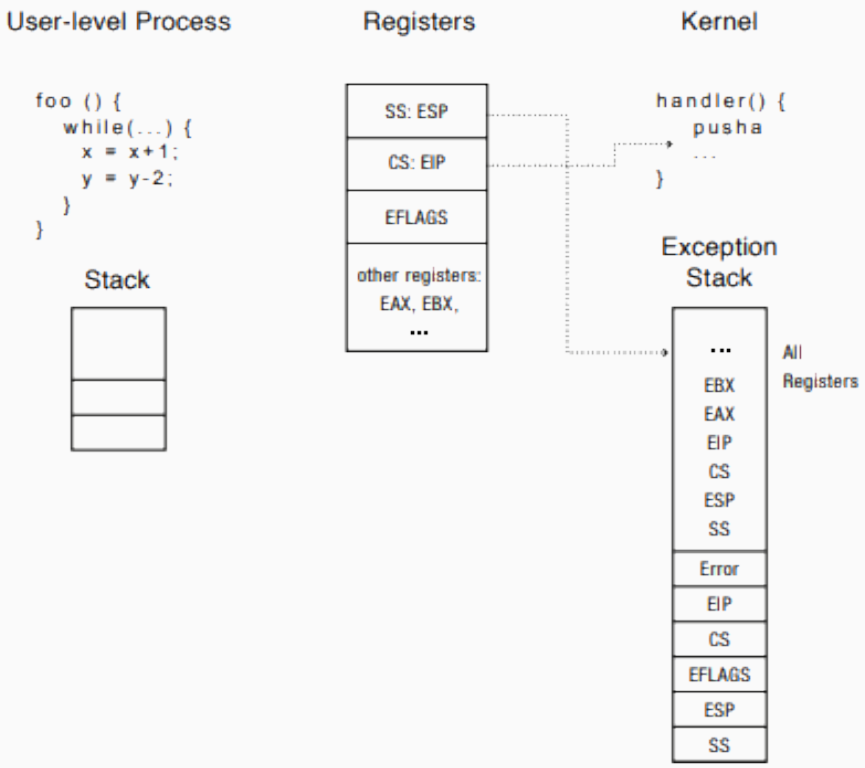


Figure 7: Gestionador comienza la ejecución

Modo Dual de Ejecución

Organización - Modo Dual de Ejecución

- Hemos visto que existen instrucciones que se deben ejecutar y que no son parte de un proceso de usuario.
- Para proteger el SO se otorgan privilegios de ejecución.
- El SO dispone de dos modos de ejecución: **Kernel** y **Usuario**.
- Modo Kernel cuenta con todos los privilegios del HW.
- Requiere soprte de HW: **Bit de modo** (0 para el Kernel)

Organización - Modo Dual de Ejecución

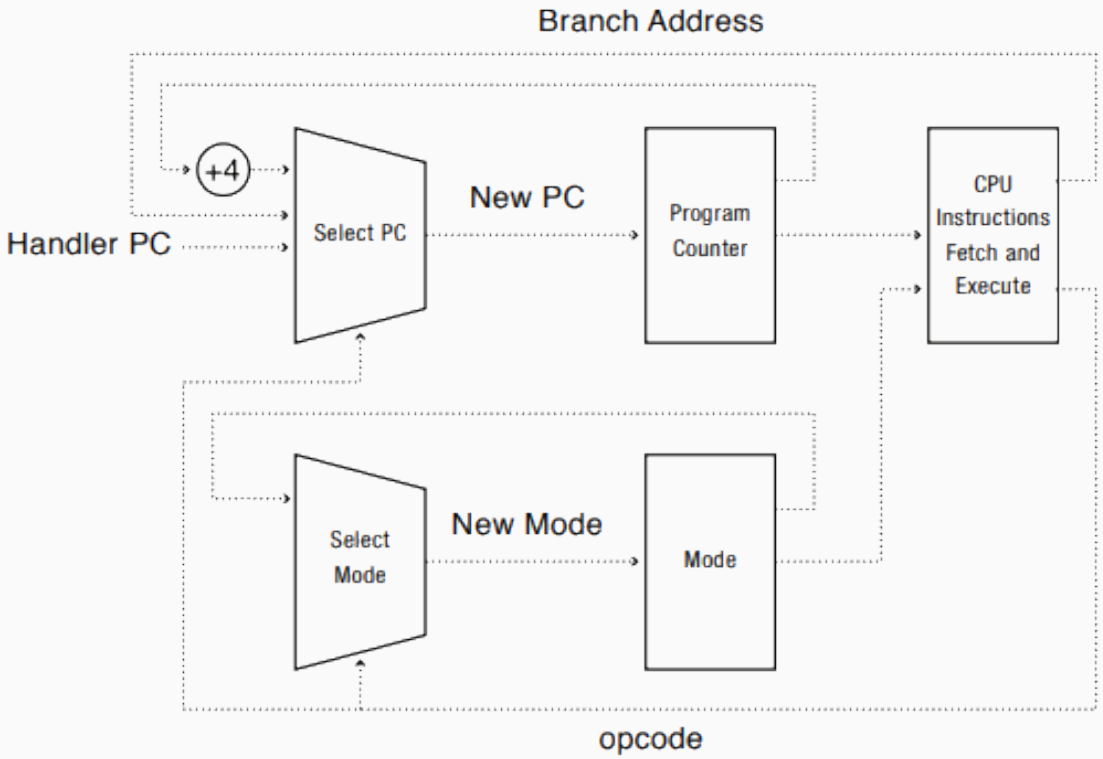


Figure 8: Vista macro del modo dual

Organización - Modo Dual de Ejecución

¿Qué produce un cambio de modo **usuario** a **kernel**?

- Una interrupción producida por el timer o E/S.
- Una excepción producida por un comportamiento no esperado en la ejecución de un proceso.
- Uso de una llamada al sistema.
  - ✓ Solicitud de un proceso para utilizar un servicio del SO.
  - ✓ Son limitadas y están cuidadosamente escritas.
  - ✓ Solo se pueden ejecutar en modo kernel.

Organización - Modo Dual de Ejecución

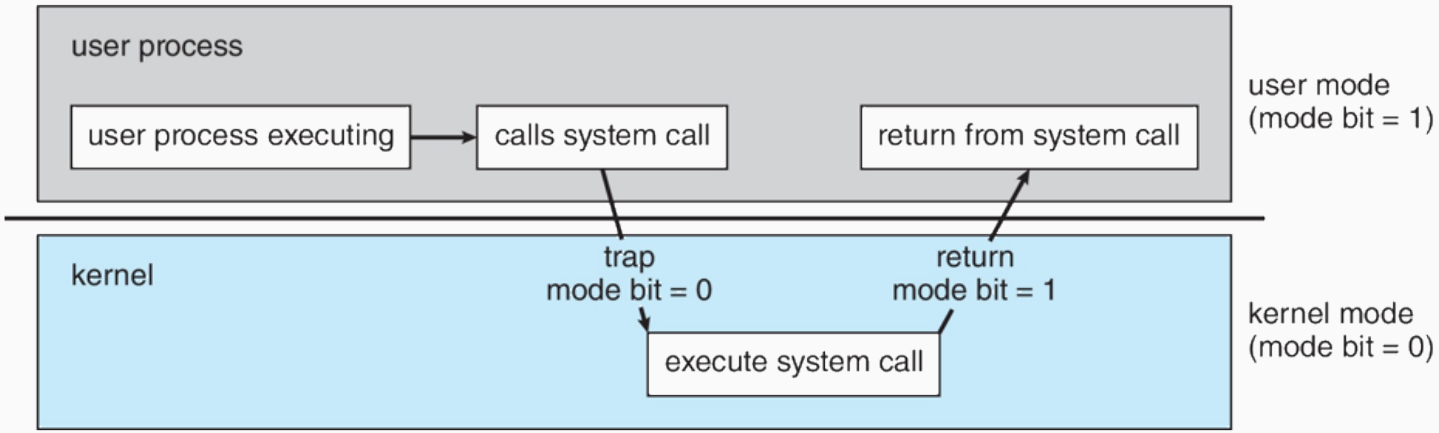


Figure 9: Cambio Modo Dual

Organización - Modo Dual de Ejecución

¿Qué produce un cambio de modo **kernel** a **usuario**?

- Inicio de un nuevo proceso (salto a la primera dirección).
- Retorno desde un evento de interrupción.
- Cuando se cambia de un proceso a otro.
- Mensajería asíncrona.
  - ✓ Procesos pueden recibir notificaciones de eventos (Upcall).

Organización - Modo Dual de Ejecución

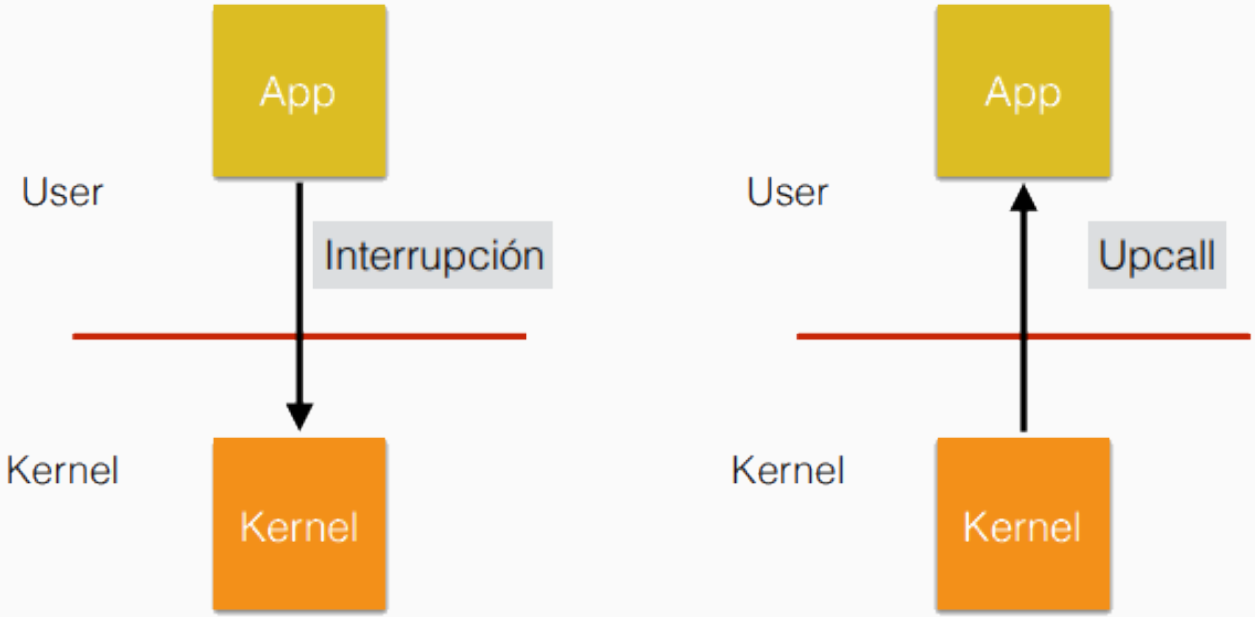


Figure 10: Upcalls

Organización - Modo Dual de Ejecución

¿Qué son las **Upcalls**?

- Interrupción por software o a nivel de usuario.
- Señales en UNIX, eventos asíncronos en Windows.
- Mecanismo de notificación a procesos de usuario sobre algún evento que necesita ser gestionado.
- El compartamiento es similar a una interrupción del kernel.
- La idea es permitir que una aplicación responda a interrupciones.
- UNIX las utiliza para notificar que algún evento a ocurrido.
- Se utiliza un Signal handler.

Llamadas al Sistema

Organización - Llamadas al Sistema

- Interfaz de programación que permite acceder a los servicios del SO.
- Se accede a ellas a través de una **API**.
- Tienen un id único para ser identificadas.
- El SO las mantiene en una tabla indexada por el id.
- La interfaz invoca la llamada y entrega el resultado.
- Quien llama al servicio no necesita saber como está hecho.
- La API oculta detalles de implementación.

Organización - Llamadas al Sistema

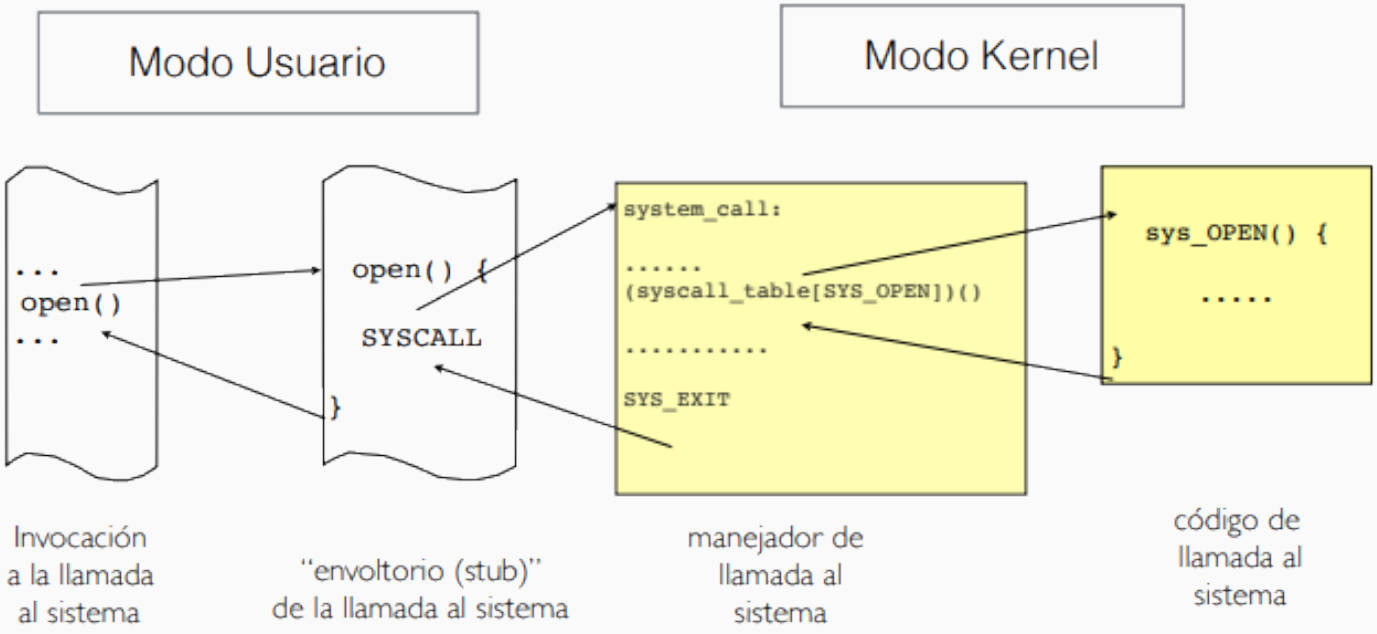


Figure 11: Cambio de Modo

Organización - Llamadas al Sistema

El kernel dispondrá de un manejador de llamadas al sistema.

- Localizar argumentos (registros o stack)
- Copia los argumentos en memoria del kernel.
- Valida los valores (protección).
- Copia los resultados en memoria de usuario.

Organización - Llamadas al Sistema

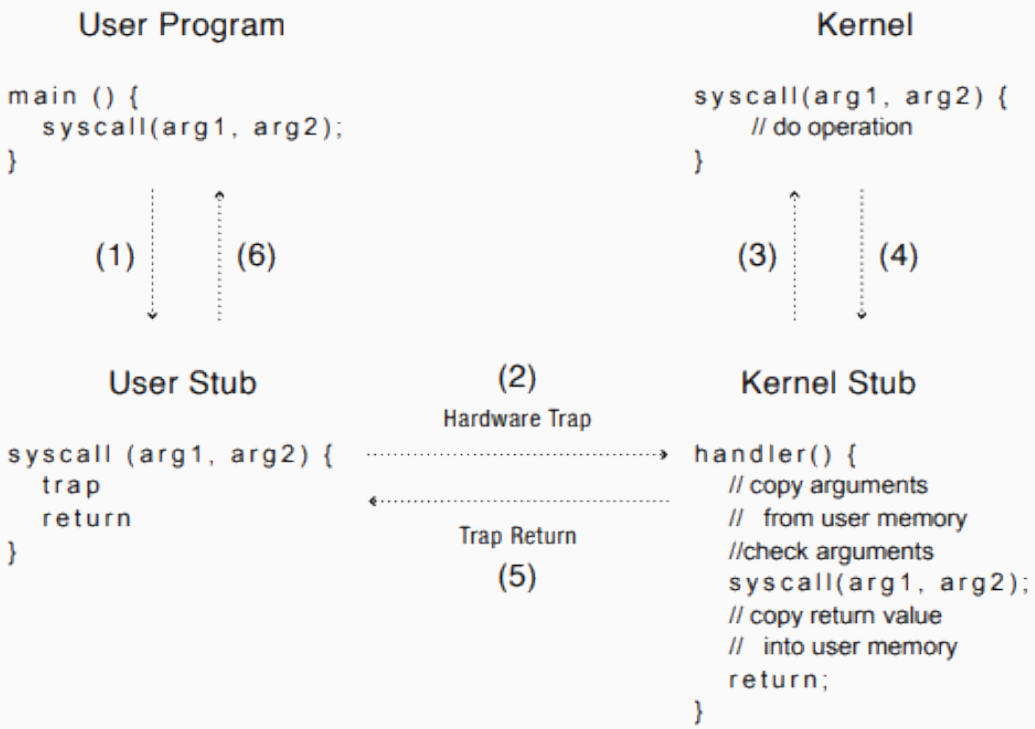


Figure 12: Eventos

Organización - Llamadas al Sistema

3 formas para implementar el paso de parámetros:

- En registros. Muchos parámetros es un problema.
- En memoria. Se pasa en un registro la dirección.
- Stack: El programa carga y el SO extrae.

Organización - Llamadas al Sistema

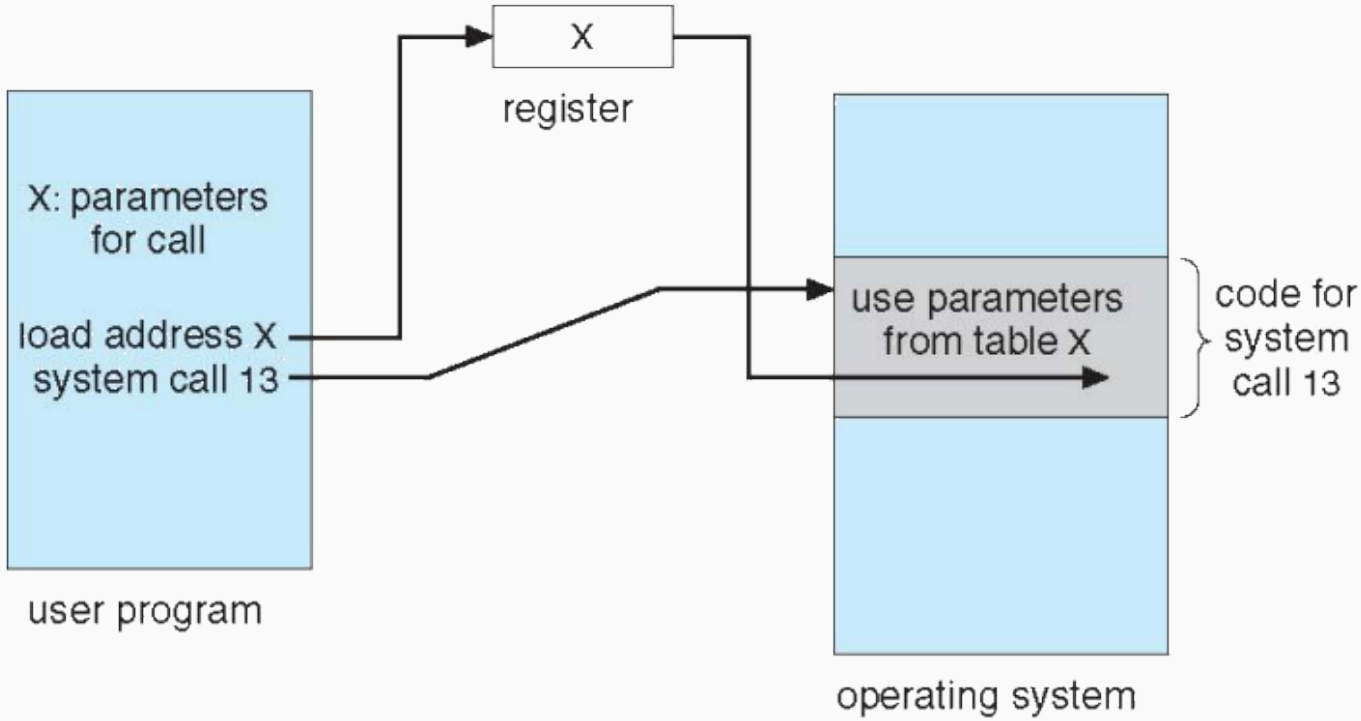


Figure 13: Paso de parámetros en las llamadas al sistema

Organización - Llamadas al Sistema

Se pueden clasificar según el servicio del SO que invocan:

- Gestión de procesos.
- Gestión de Archivos.
- Gestión de E/S.
- Contabilidad.
- Comunicación.
- Protección.

Servicios Estructurados

Diseño - Servicios Estructurados

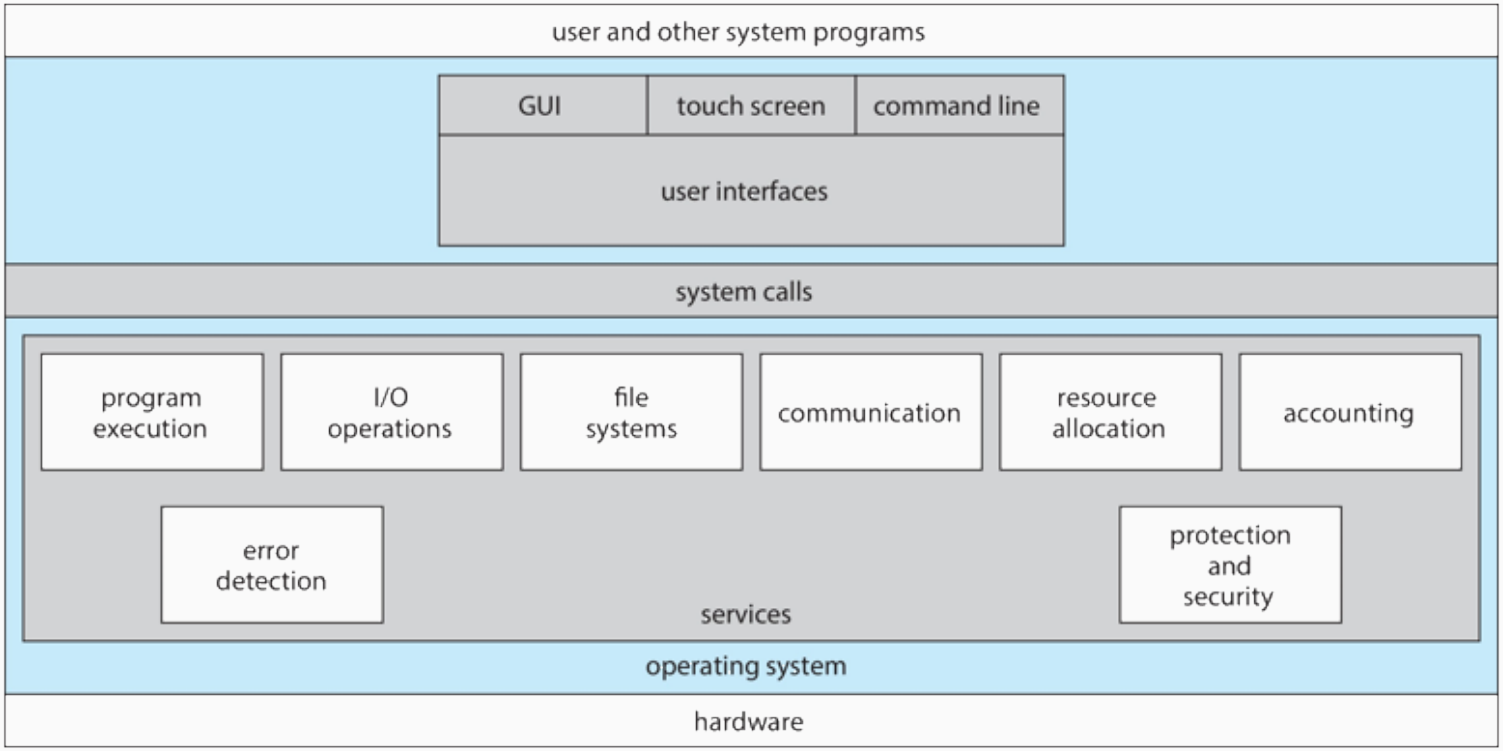


Figure 14: Servicios del SO

Diseño - Servicios Estructurados

La interfaz de usuario es parte del grupo de servicios:

- Línea de comando (CLI).
- Interfaz gráfica (GUI).
- Pantallas táctiles.
- Batch.

Diseño - Servicios Estructurados

SO's modernos obedecen dos formas de estructurar los servicios:

- Kernel monolíticos.
- Micro Kernel.

Diseño - Servicios Estructurados

Kernel Monolítico

- Windows y Linux siguen, en parte, esta estructura.
- La mayoría de los servicios son parte del kernel.
- La portabilidad dependerá exclusivamente de:
  - ✓ Capa de abstracción de HW (HAL).
  - ✓ Instalación dinámica de drivers.

Diseño - Servicios Estructurados

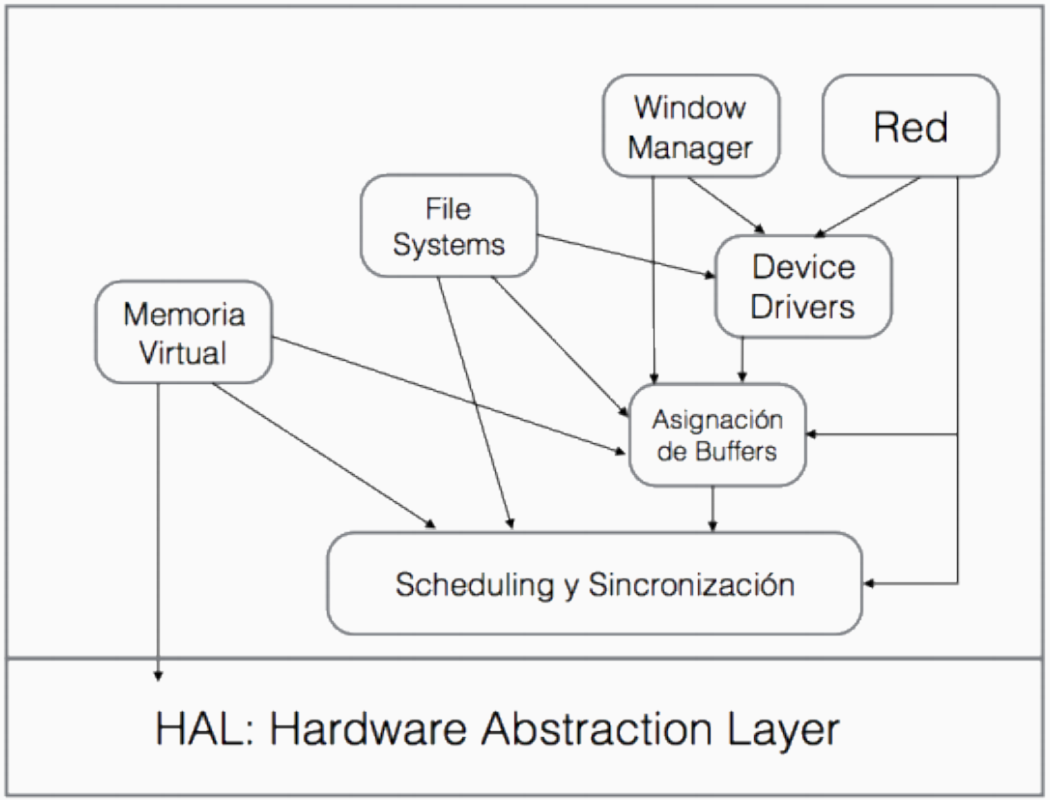


Figure 15: Estructura Monolítica

Diseño - Servicios Estructurados

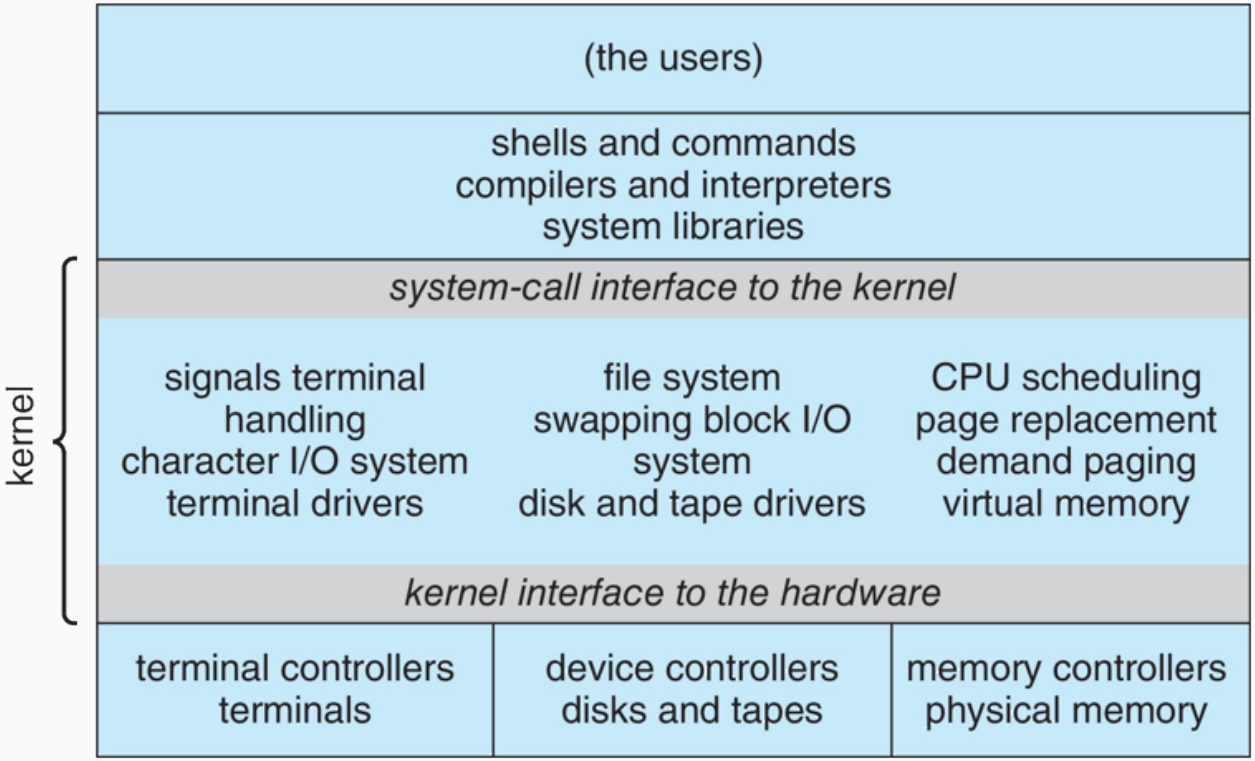


Figure 16: Estructura Unix

Diseño - Servicios Estructurados

HAL

- Portabilidad del SO sobre distintas arquitecturas de HW.
- Es una interfaz que contiene configuraciones y operaciones específicas de procesadores.
- Una capa bien definida permite SO's independientes del procesador.
- Portar un SO significa modificar rutinas de bajo nivel de HAL.

Drivers

- Un SO necesita instalar gran variedad de E/S.
- Existe gran variedad de interfaces de HW para E/S.
- Un 70% del código del kernel de Linux es para E/S.
- Drivers de carga dinámica permiten manejar los E/S.
- Cuando el SO inicia busca los E/S conectados y carga los drivers.
- El SO puede cargar los drivers desde internet.

Kernel Monolítico

- Las interfaces de comunicación entre distintas partes del SO generalmente son complejas.
- Esto lleva a que sea propenso a errores. Un error es fatal.
- Si se produce un error en modo kernel, el kernel fallará.
- Será necesario reiniciar el computador para volver a comenzar.

Micro Kernel

- Para minizar el riesgo anterior se utiliza esta estructura.
- La idea es mover todo lo posible al espacio usuario.
- Se debe implementar un mecanismo de comunicación.
- No existe una definición formal de qué elementos van en el kernel.
- Generalmente: Gestión de memoria, procesos y comunicación.
- Mac OS X kernel (Darwin) basado en Mach es un ejemplo.

Diseño - Servicios Estructurados

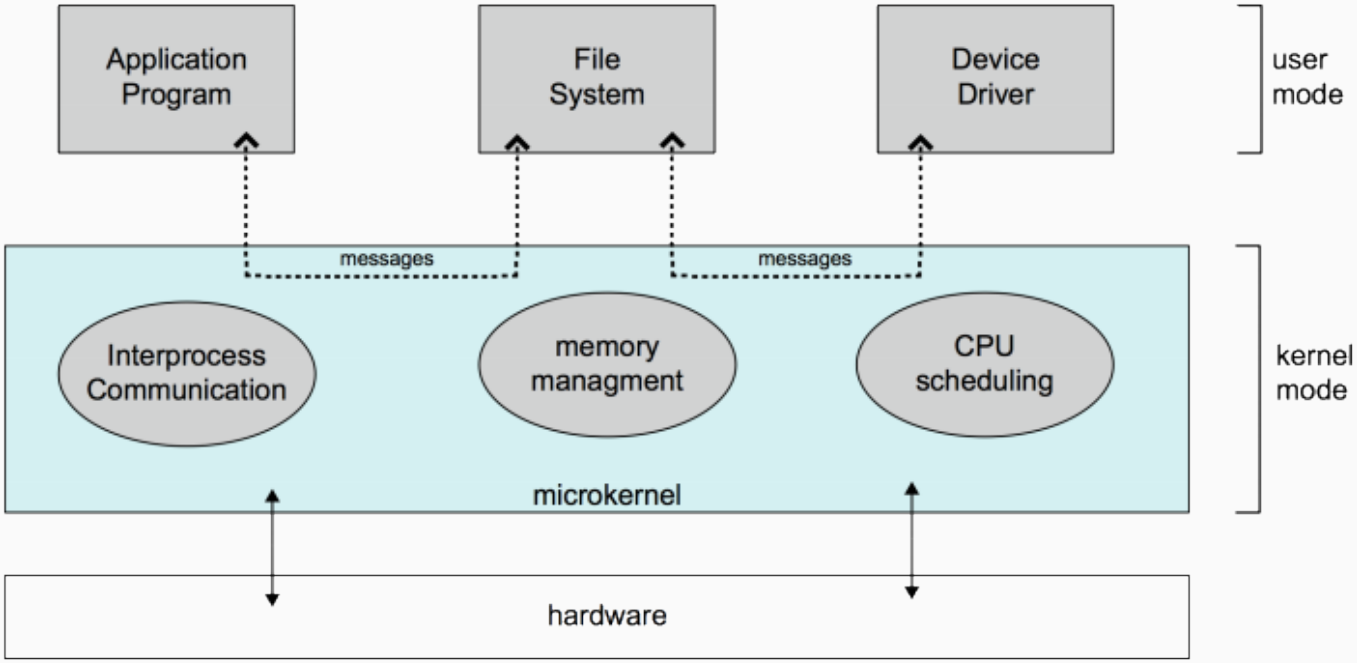


Figure 17: Micro Kernel

Micro Kernel

- Beneficios.
  - ✓ Facil de extender.
  - ✓ Portable.
  - ✓ Más confiable y seguro.
- Problema: Overhead producto de la comunicación entre módulos del kernel y usuario.

Diseño - Servicios Estructurados

Sistemas Híbridos: SO's modernos mezclan lo mejor de ambos mundos:

- Linux y Solaris usan kernel monolítico junto a carga dinámica de módulos.
- Windows en su mayoría monolítico pero con micro kernel para personalización del sistema.
- Mac OS X híbrido, por capas, Aqua UI junto al ambiente de desarrollo Cocoa.
  - ✓ Kernel basado en Mach microkernel.
  - ✓ Componentes de BSD Unix .
  - ✓ Gestión de E/S y carga dinámica de módulos.

Diseño - Servicios Estructurados

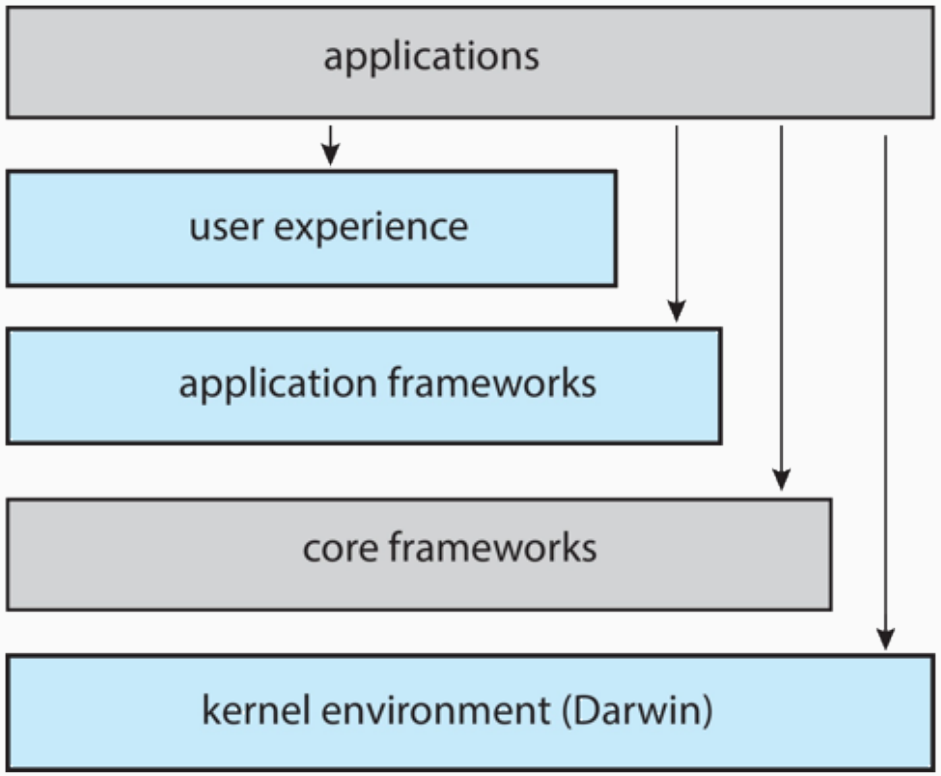


Figure 18: Mac OS & iOS

Diseño - Servicios Estructurados

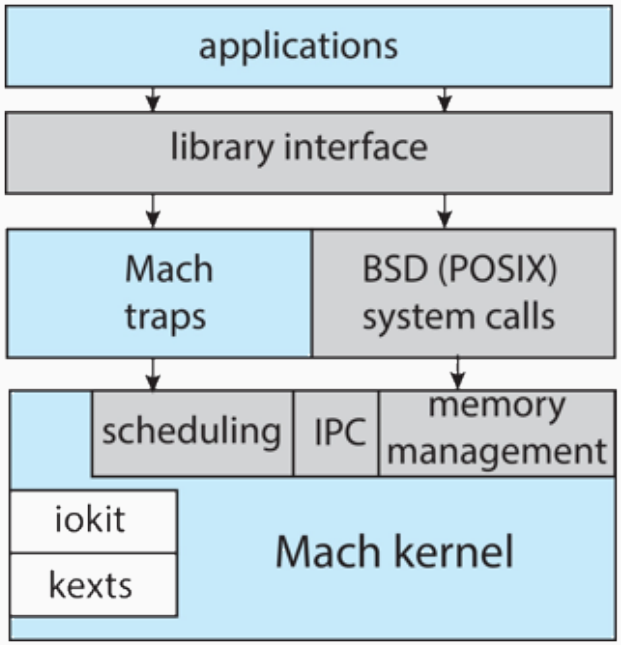


Figure 19: Darwin

Diseño - Servicios Estructurados

La forma de estructurar los servicios persigue dos objetivos:

- **Usuario:** Seguridad, confiabilidad, desempeño, usabilidad.
- **Sistema:** Simple, flexible, portable, libre de errores.

Hint de diseño

Separar las políticas de los mecanismos. ¿Por qué? Las políticas pueden cambiar en el transcurso del tiempo.

Virtualización

Diseño - Virtualización

- Ejecutar más de un SO de forma concurrente en un computador.
- Componentes:
  - ✓ **Host:** SO nativo del computador.
  - ✓ **VVM:** Gestionador de VM's.
  - ✓ **Guest:** SO instalado en la VVM.

Diseño - Virtualización

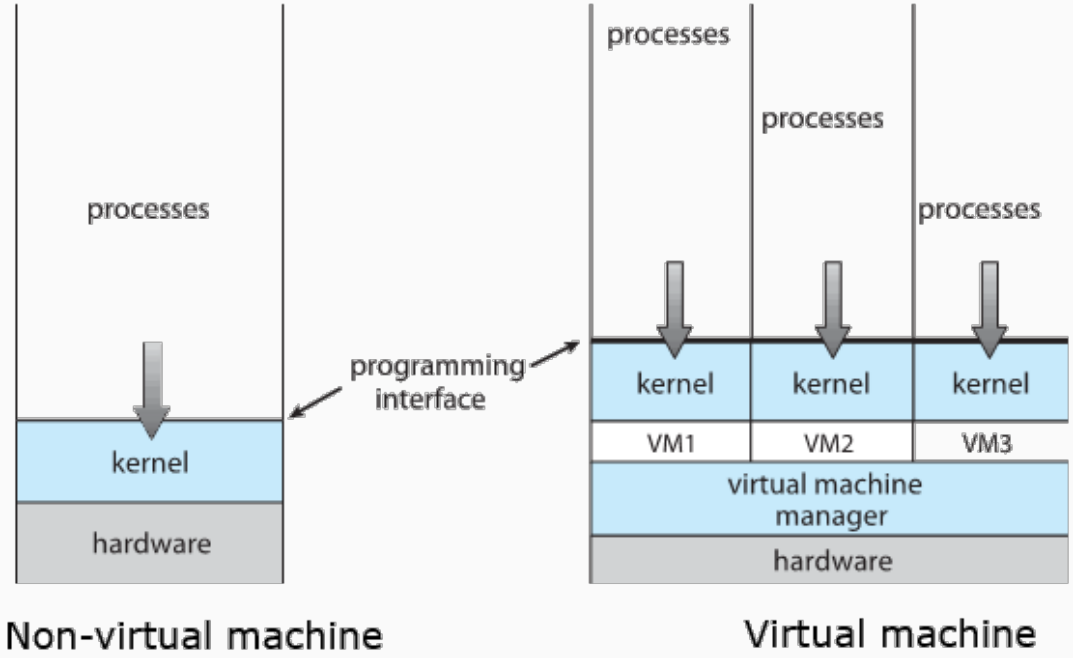


Figure 20: Máquina Virtual

Diseño - Virtualización - Implementación VMMs

- Tipo 0: Basado en hardware vía firmware.
- Tipo 1: SO entrega la funcionalidad.
  - ✓ Microsoft Windows Server con HyperV.
  - ✓ RedHat Linux con KVM.
- Tipo 2: Aplicación entrega la funcionalidad.
  - ✓ VMWare.
  - ✓ Parallels.
  - ✓ Oracle VirtualBox.

Diseño - Virtualización - Implementación VMMs

- ¿Cómo lograrlo? ¿Qué se debería hacer?
- Simular:
  - ✓ Inicio del sistema.
  - ✓ Gestión de Interrupciones.
  - ✓ Modo dual de ejecución.
  - ✓ Llamadas al sistema.
  - ✓ Planificación de uso de CPU.
  - ✓ Gestión de E/S.
  - ✓ Etc.

Para el inicio:

- El host carga el *bootloader* del guest desde el disco virtual.
- El bootloader carga el kernel del guest y comienza a ejecutarse.
- El guest inicia su vector de interrupciones.
- El guest carga un proceso desde el disco virtual a la memoria.

Diseño - Virtualización - Implementación VMMs

Para el modo dual:

- CPU virtual para representar el estado de las visitas.
- El kernel de la visita no debe ejecutarse en modo kernel del host.
- Se implementan dos modos virtuales (Usuario / Kernel).
- Pero ambos se ejecutan en modo usuario del host.

Diseño - Virtualización - Implementación VMMs

Para las interrupciones:

- La VMM gestiona, resuelve y retorna.
- Las instrucciones privilegiadas suelen ser más lentas.
- Sobre todo cuando tenemos múltiples sistemas.

Diseño - Virtualización

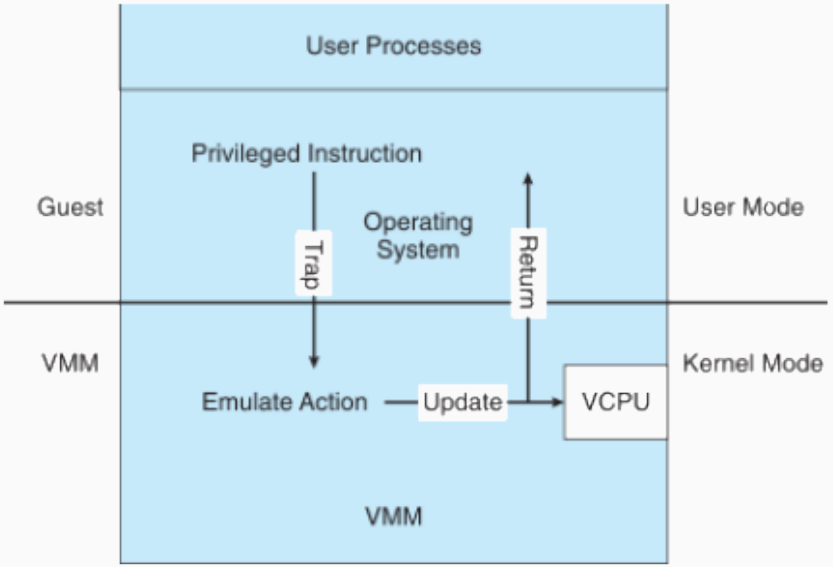


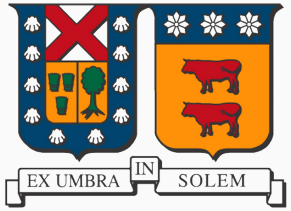
Figure 21: Interrupciones virtualizadas

Diseño - Virtualización

Beneficios

- Múltiples SO's corriendo en el mismo HW.
- Ejecución de procesos de forma concurrente en distintas plataformas.
- Ahorro de recursos de hardware.
- Protección entre el host y sus guest.
- Gestión de estados: Congelar, suspender, ejecutar y clonar.
- Templates: SO + App's.
- Migración en línea: Mover una MV en ejecución entre host's.

Todos estos beneficios juntos $\rightarrow$ Cloud Computing.



Sistemas Operativos

Organización y Diseño

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.