Sistemas operativos avanzados

Tema 1

Introducción

Contenido

- *Definición de SO*
- *Componentes del SO*
- *Estructura del SO*
- *Principios de diseño del SO*

Definición de Sistema Operativo (*déjà vu*)

- "Ubicuos pero no hacen un trabajo útil concreto"
- Gestión segura y eficiente de los recursos del computador
  - Reparto temporal y espacial de recursos entre programas
- Ofreciendo abstracción de "máquina extendida"
  - Servicios que ocultan e independizan del hardware
    - Crean abstracciones de recursos hardware
- ¿Algo más?: Sólo con eso, tan inútil como máquina desnuda
  - SO de propósito general debe ofrecer interfaz a usuarios
    - ¿Debe considerarse interfaz como parte del SO?
    - Suele estar implementado como aplicación externa
  - Definición precisa de SO: Asunto polémico
    - Linux vs. GNU/Linux
    - Juicio antimonopolio a Microsoft por IE
  - ¿A qué nos referimos cuando hablamos de SO?

Distintas interpretaciones del término SO

- Estricta (la que usaremos en la asignatura):
  - Gestor de recursos que ofrece servicios a aplicaciones
    - Interfaz de usuario es otra aplicación más fuera del SO
  - SO = Núcleo (*kernel*): software que ejecuta en modo sistema
    - No aplicable a sistemas con arquitectura micronúcleo
      - Parte del SO ejecuta en modo usuario
- Amplia (concepto de distribución en Linux):
  - Todo el software que hace operativo al sistema
    - Software de interfaz de usuario (p.e. GUI, *bash*)
    - Herramientas del sistema (p.e. montador *ld*)
    - "Demonios del sistema" (p.e. *spooler* de impresora)
    - Bibliotecas del sistema (p.e. *libc*)

Componentes del sistema operativo

- Gestión de procesos
- Gestión de memoria
- Sistema de entrada/salida
- Sistema de ficheros
- Sistema de seguridad y protección

Gestión de procesos

- Abstracción fundamental del SO
  - Proceso: programa en ejecución
- Cada proceso tiene un conjunto de recursos asociados:
  - Flujos de ejecución internos (*threads*)
  - Mapa de memoria
  - Ficheros abiertos, puertos de comunicación, ...
- SO debe controlar:
  - Creación y destrucción de procesos
  - Comunicación y sincronización del proceso
    - Así como asegurar su propia sincronización interna
  - Asignación y liberación de recursos al proceso
    - Evitando los **interbloqueos**
  - **Planificación de UCP**: qué proceso ejecuta en cada instante

Gestión de memoria

- SO ofrece espacio lógico propio (mapa) a cada proceso
  - Mapa del proceso incluye todas las regiones requeridas
    - Código, datos, pilas, DLL, ficheros proyectados, etc.
- SO gestiona mem. de sistema usando esquema fijado por MMU
  - Registros base/límite, segmentación, paginación, ...
- SO implementa la técnica de memoria virtual que permite:
  - Ejecutar procesos cuyo mapa es más grande que la memoria
  - Aumentar el grado de multiprogramación

Sistema de entrada/salida

- Manejadores se encargan de gestionar los dispositivos
  - Uno por cada tipo de dispositivo
  - Ofrecen interfaz común a pesar de gran variedad de dispositivos
  - Gestionan todos los aspectos hardware (p.e. DMA)
- Implementación de aspectos avanzados
  - Control de consumo de energía del dispositivo en portátiles
  - *Hot-plugging*
- Manejador de disco crítico en SO (sirve a G. memoria y S. Fich.)
  - Algoritmos de planificación del disco

Sistema de ficheros

- Fichero: abstracción de espacio de almacenamiento
- Espacio jerárquico de nombres usando directorios
- Tendencia: dar soporte a distintos tipos de sistemas de ficheros
  - Concepto de sistema de ficheros virtual: interfaz común
- Cada tipo de sistema de ficheros específico usa estrategias para:
  - Organización del espacio ocupado por los ficheros
  - Gestión del espacio libre
  - Técnicas de prevención ante caídas (p.e. *journaling*)

Sistema de seguridad y protección

- Protección:
  - ¿Qué operaciones puede hacer usuario con recurso?
  - ¿Cómo almacenar información sobre permisos?
    - Asociada al usuario: Capacidades
    - Asociada al recurso: Listas de control de acceso
- Autenticación:
  - Asegurar que un usuario es quien dice ser
- SO debe evitar amenazas a la seguridad e integridad del sistema

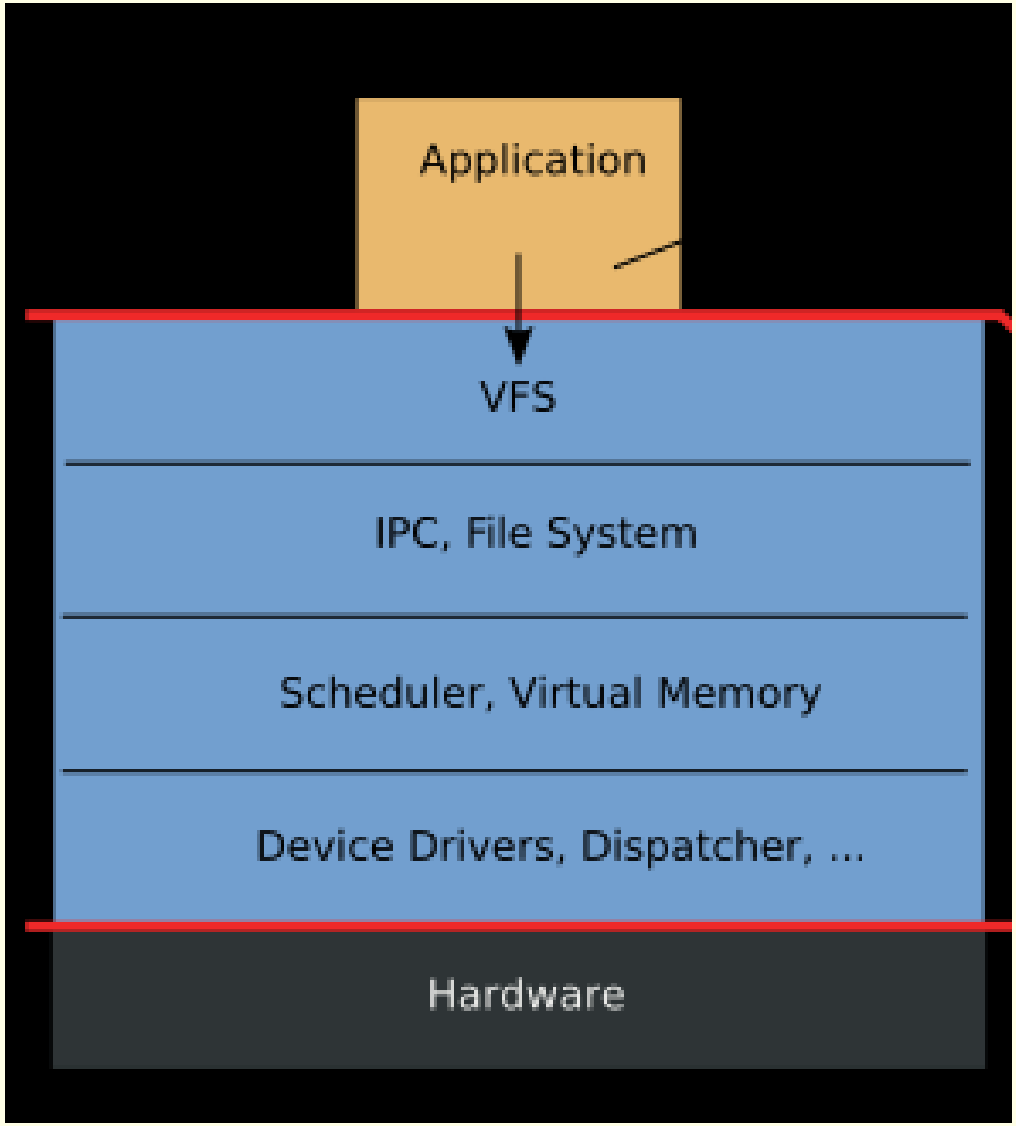
Estructura del sistema operativo

- Existen distintas alternativas:
  - Sistemas monolíticos
  - Sistemas por capas
  - Sistemas basados en micronúcleos
  - Sistemas híbridos
  - Exonúcleos
- Máquinas virtuales

Sistemas operativos monolíticos

- SO = núcleo (*kernel*) → programa que ejecuta en modo sistema
  - Todo el código del SO enlazado en un único programa que
  - Ejecuta en un mismo espacio de direcciones
- Aplicaciones y programas de sistema ejecutan en modo usuario
- Ventaja: Eficiencia
  - Ejecución de servicio de SO:
    - Sólo requiere cambio de usuario a sistema y viceversa
- Desventaja: Difícil depuración y extensibilidad
  - Error en cualquier parte del SO afecta al resto
- Es la arquitectura más habitual
  - Característica de la familia UNIX (Linux)

Sistema monolítico (wikipedia)



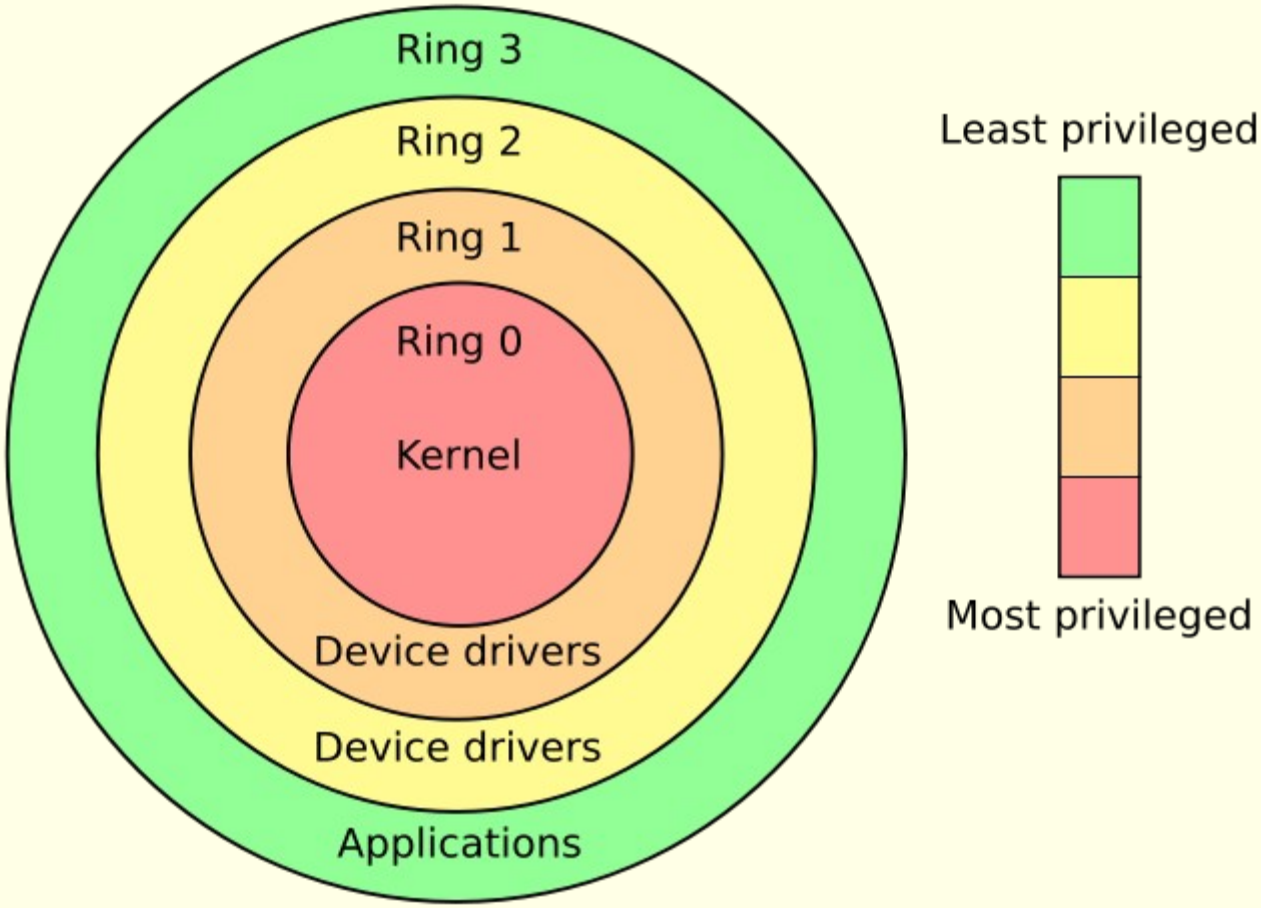
Sistemas con módulos cargables

- Mayoría de sistemas monolíticos actuales no están “cerrados”
  - Permiten cargar módulos en tiempo de ejecución
- Ejecutable del SO contiene funcionalidad básica
- Restante en módulos (manejadores, s. ficheros, protocolos, etc.)
- Módulo similar a biblioteca dinámica pero para el núcleo
  - Se incorpora a espacio de SO y comparte símbolos
  - Se mantienen los mismos problemas de fiabilidad
- Ventajas:
  - Facilita extensibilidad del SO
  - Permite adaptar núcleo a características de la plataforma
    - P.ej. Crear núcleo mínimo para sistema empotrado
  - Posibilita técnicas como *hot-plugging*

Sistemas por capas (o anillos)

- Organizar funcionalidad del SO en capas independientes
  - Cada capa es un ejecutable independiente que
  - Ejecuta en su propio espacio de direcciones
  - Organizadas en niveles de mayor a menor privilegio
  - Capa sólo se comunica con adyacentes usando llam. al sistema
    - Procesador verifica que se trata de capas adyacentes
- Facilita depuración y controla propagación de errores
- Requiere que procesador provea de varios niveles de privilegio
  - Pentium proporciona 4
  - S. monolíticos sólo requieren dos modos (usuario/sistema)
    - Más transportables
- Primer SO por capas: THE (Dijkstra, 1968)
  - MULTICS también usó esta arquitectura

Sistema organizado en anillos (wikipedia)



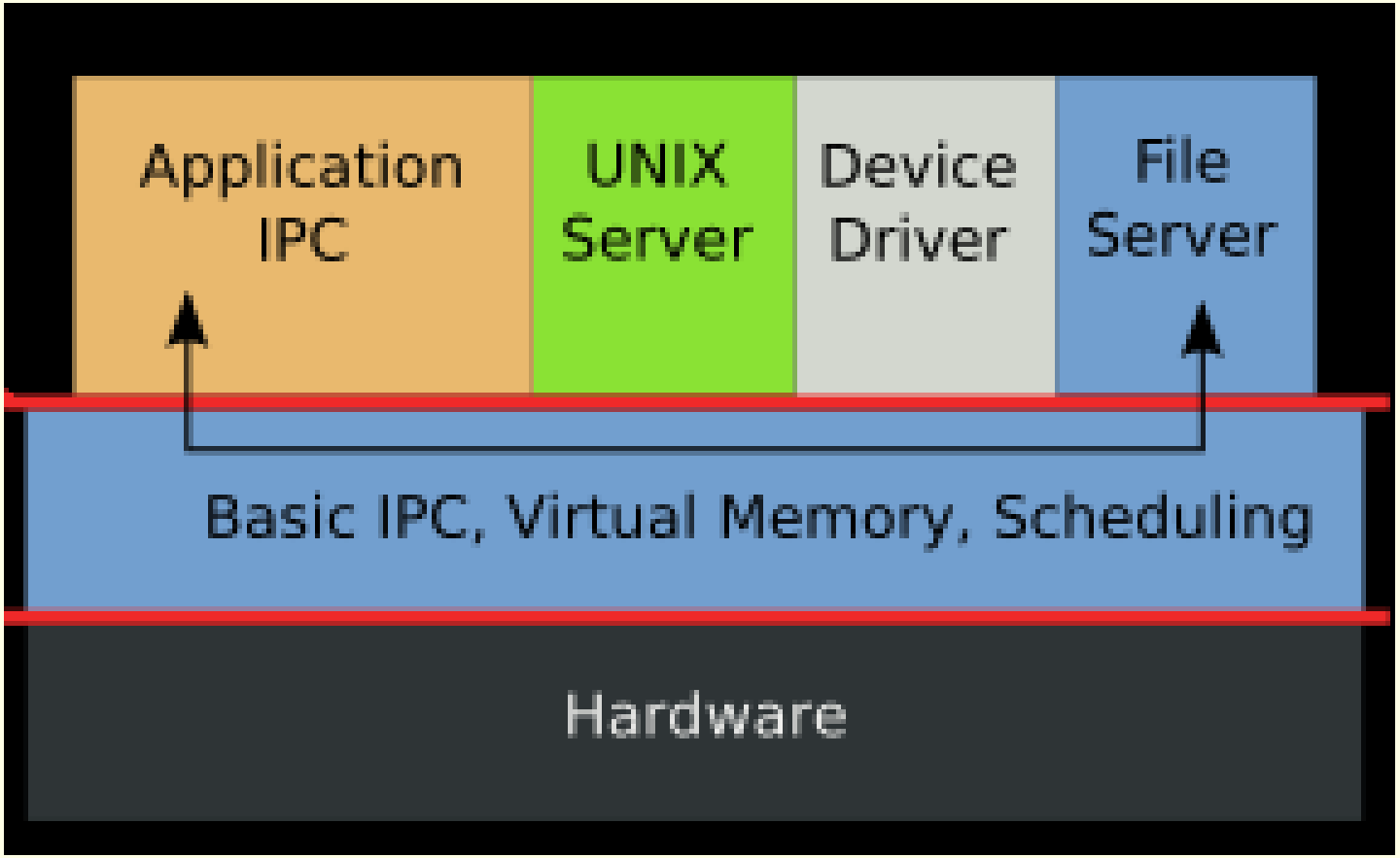
Micronúcleo (I)

- "Lo perfecto no es que no falte nada sino que no sobre nada"
- Núcleo queda reducido a funcionalidad mínima
  - Gestión de procesos y de memoria de bajo nivel + IPC
    - Micronúcleo proporciona nº muy reducido de llamadas
- Funcionalidad del SO en servidores en modo usuario
  - Sistema de ficheros, gestor de memoria, manejadores, ...
  - "Llamada al sistema" de aplicación: mensaje a servidor
    - Y entre servidores si es necesario
- Ventaja: Extensibilidad, fiabilidad y facilidad de depuración
  - Error en parte del SO sólo afecta al servidor involucrado
  - Posible convivencia de varios SS.OO.

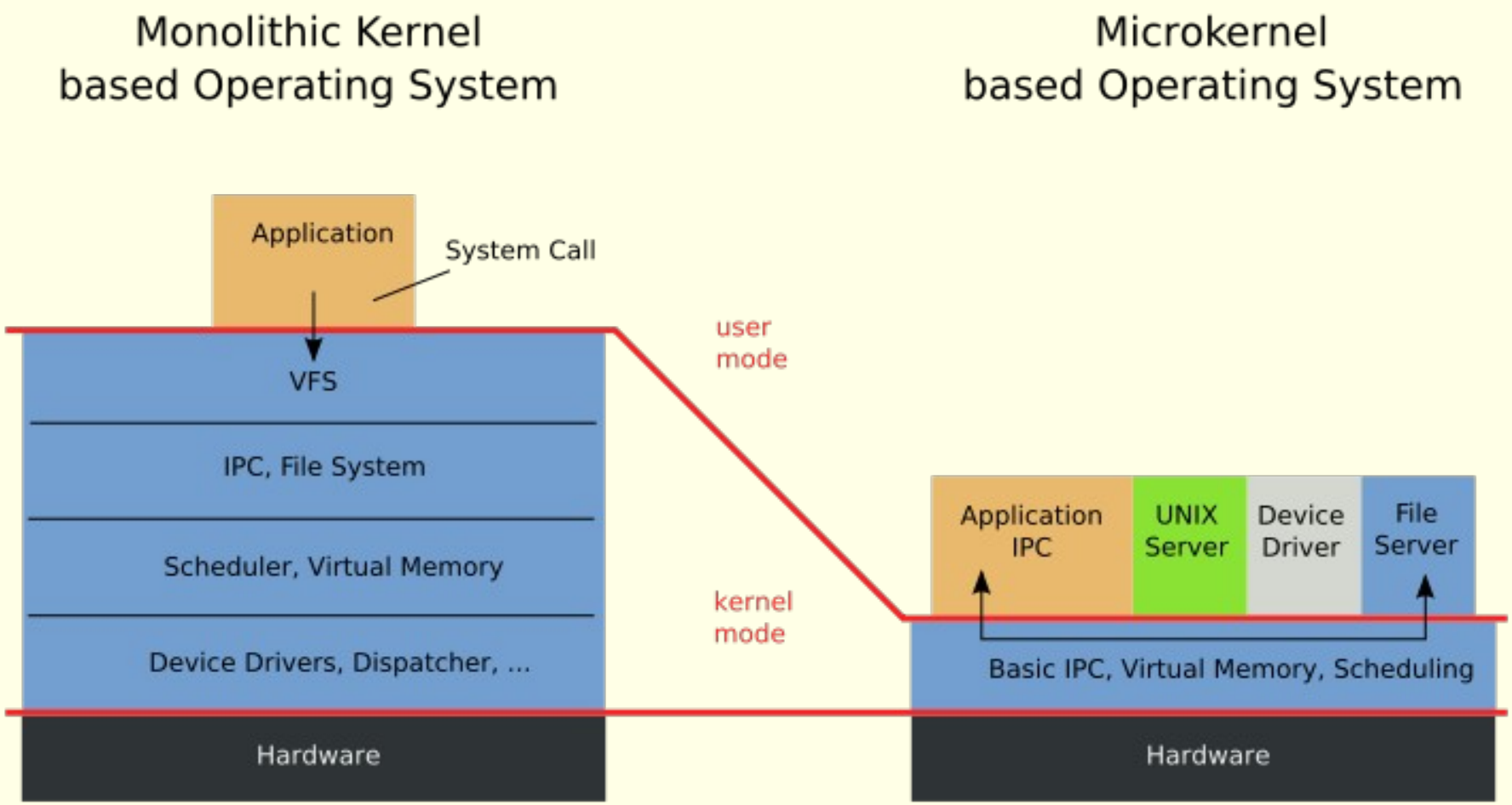
Micronúcleo (II)

- Desventaja: Eficiencia.
  - Coste de llamada: Sobrecarga mensajes y cambios de proceso
- Primeros sistemas: mono-servidor
  - Único servidor en modo usuario proporciona todos servicios
  - Menos sobrecarga por mensajes y cambios de contexto
  - Pero pierde muchas de las ventajas de micro-núcleos
- Eficiencia ha mejorado en 2ª generación (L4) frente a 1ª (Mach)
- Tamaño del núcleo ha ido disminuyendo:
  - Mach ≈ 100 llamadas; L4 ≈ 10 llamadas
  - Nanonúcleos, piconúcleos (actualmente = micronúcleos)
- 3ª generación: núcleos verificados formalmente (SeL4)

Sistema basado en micronúcleo (wikipedia)



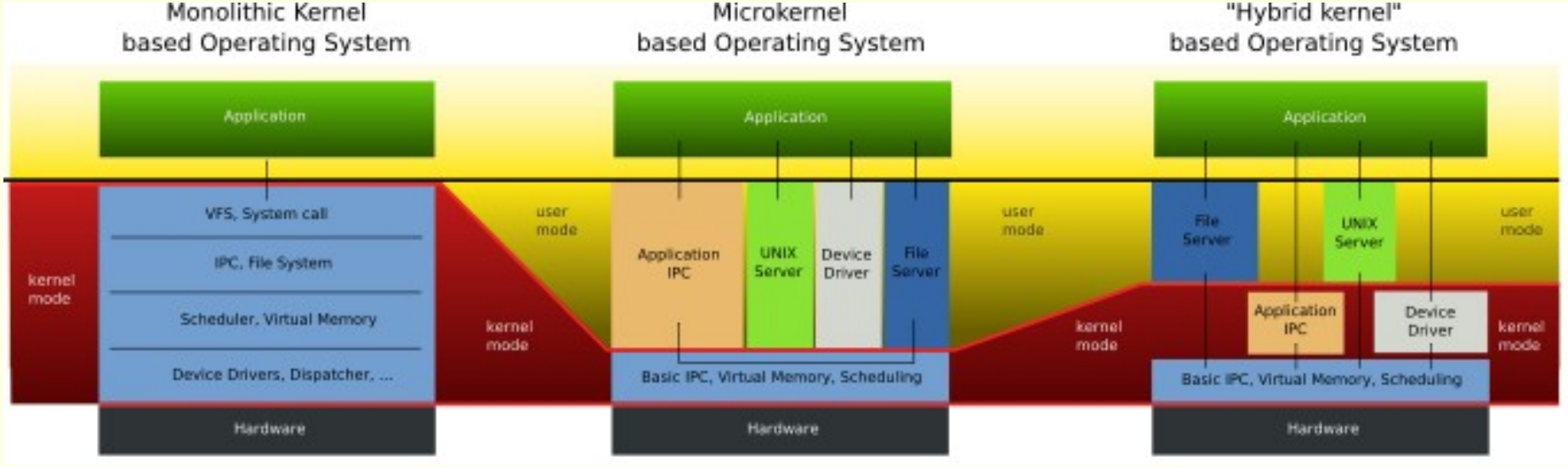
Monolítico versus micronúcleo (wikipedia)



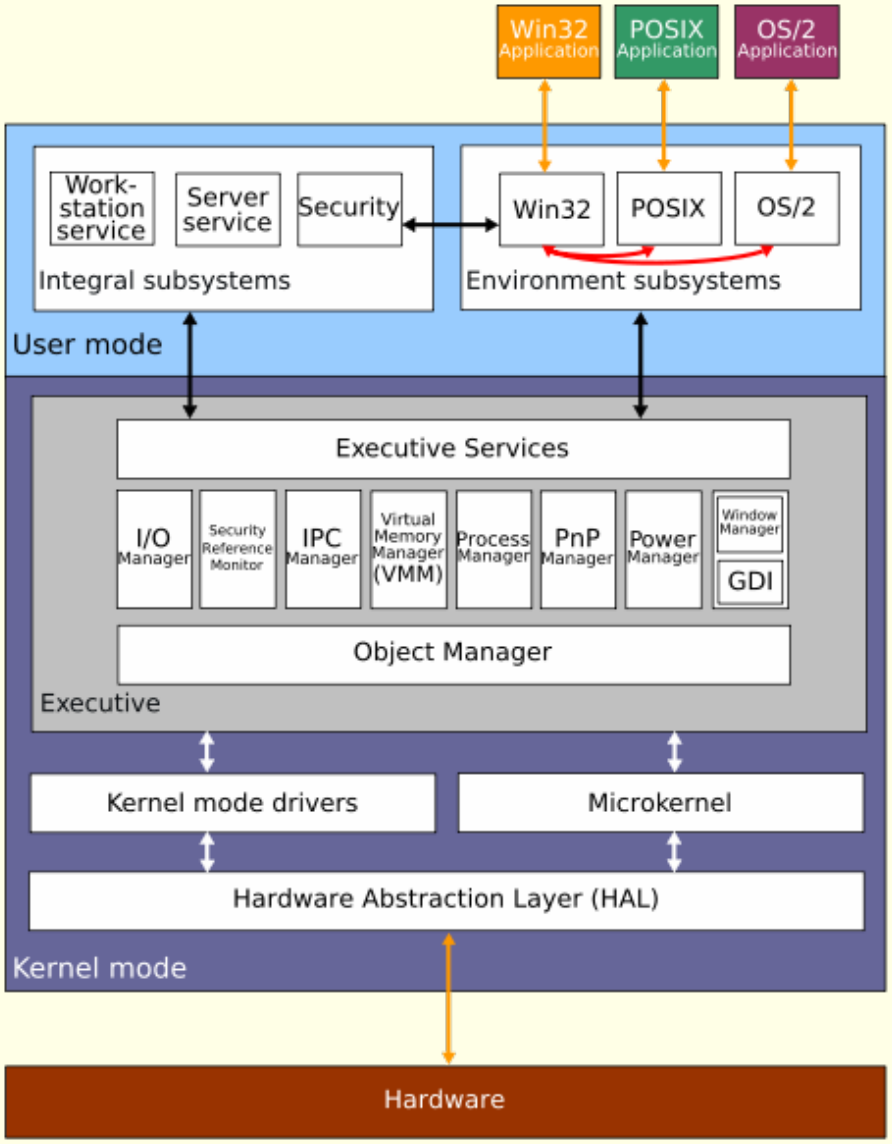
Sistemas híbridos

- Algunos s. micronúcleo permiten servidores en modo sistema
  - Más eficiente pero rompe la filosofía micronúcleo
  - Servidores son programas independientes pero
  - Ejecutan en mismo espacio de direcciones del micronúcleo
    - Y no usan IPCs para comunicarse
- Categoría discutida
  - Puristas consideran que son monolíticos
- Ejemplo: Mac OS X
  - Interfaz UNIX
  - Micronúcleo Mach con servidores en m. sistema
- ¿Y Windows?
  - Difícil clasificación: Puede considerarse híbrido

Monolítico|Micronúcleo|Híbrido (wikipedia)



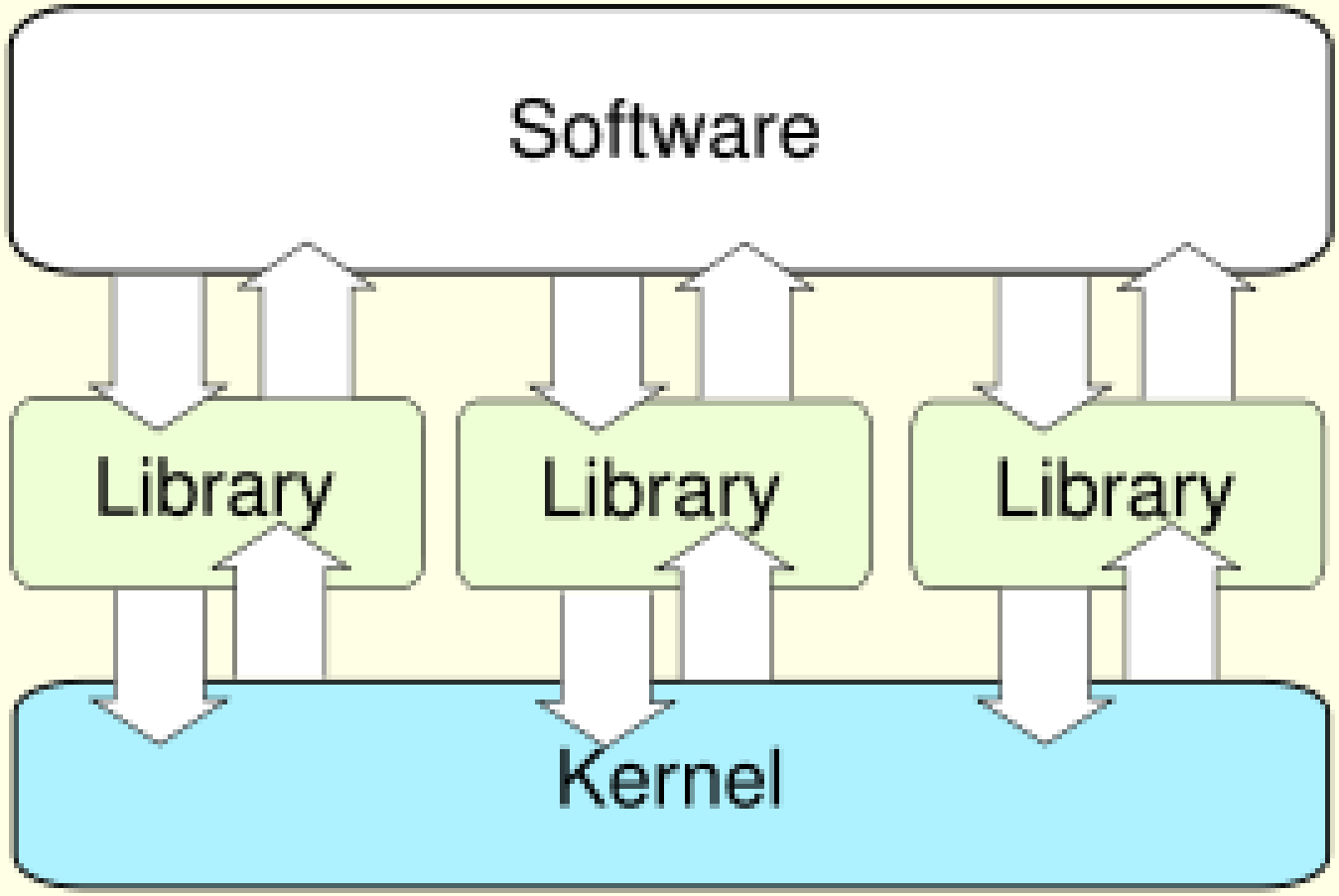
Estructura de Windows 2000 (wikipedia)



Exonúcleos

- Motivación:
  - No todas las aplicaciones necesitan ver mismas abstracciones
    - Gestor base de datos mejor maneja bloques de disco que ficheros
      - Uso de abstracciones inadecuadas es ineficiente
- Propuesta: *Exokernel* (MIT, 1995)
  - Núcleo provee abstracciones básicas (página, bloque, ...)
  - Funcionalidades de tipo SO en bibliotecas en modo usuario
  - Cada aplicación se enlaza con las bibliotecas que requiera
    - Gestor base de datos no requiere de sistema de ficheros
- Aplicación del principio "*end-to-end*"

Exonúcleo (wikipedia)



Máquinas virtuales (MV)

- Software que implementa una máquina (= o ≠ máquina real)
  - SO crea máquina virtual pero extendida (abstracción del HW)
- Pionero CP-40 (IBM, 1967): ¿t. compartido y SO monousuario?
  - CP crea 1 MV/usuario: SO monousuario CMS sobre cada MV
- Técnica de nuevo en auge actualmente
  - Capacidad de procesamiento actual palia ineficiencia de MV
  - Procesadores incluyen soporte para la misma

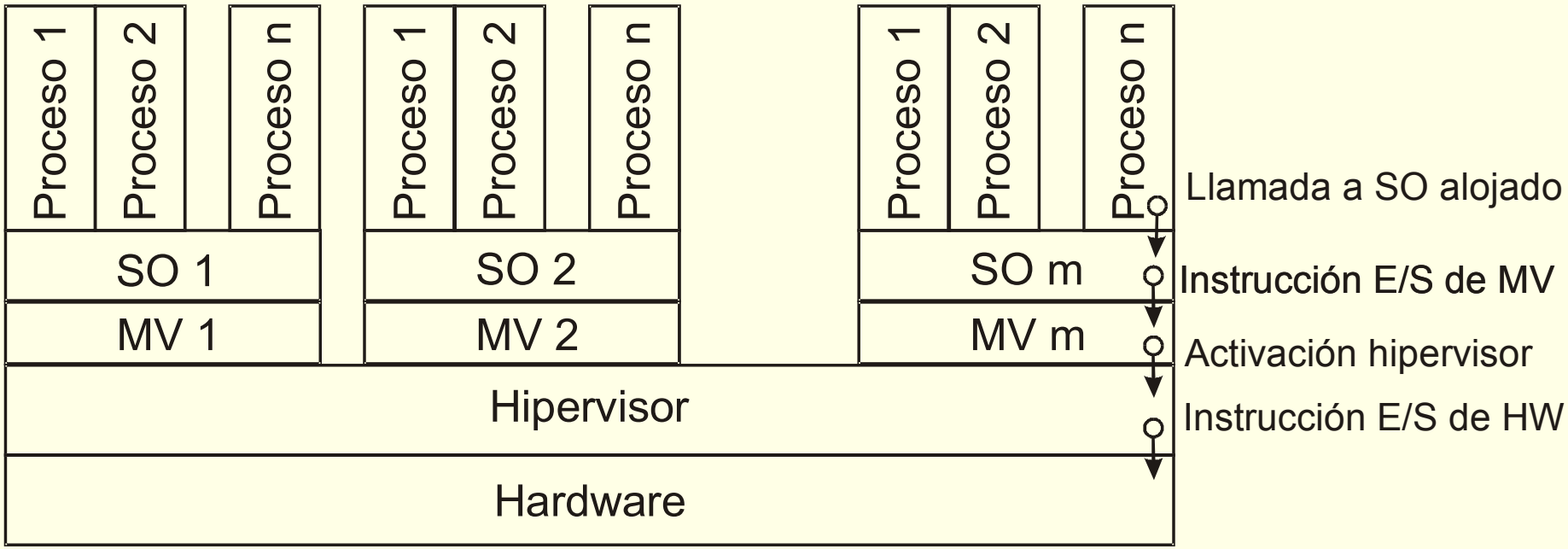
Tipos de máquinas virtuales

- Dependiendo del nivel
  - De sistema: Crea MV como soporte de ejecución de un SO
  - De proceso: Crea MV como soporte de ejecución de 1 proceso
- Mismo juego de instrucciones procesador real y virtual
  - No → emulación: intérprete versus traducción dinámica
  - Ejemplo: Bochs emula x86 mediante interpretación
- Software sobre MV consciente de su existencia
  - Sí: Paravirtualización
  - No: Plena virtualización

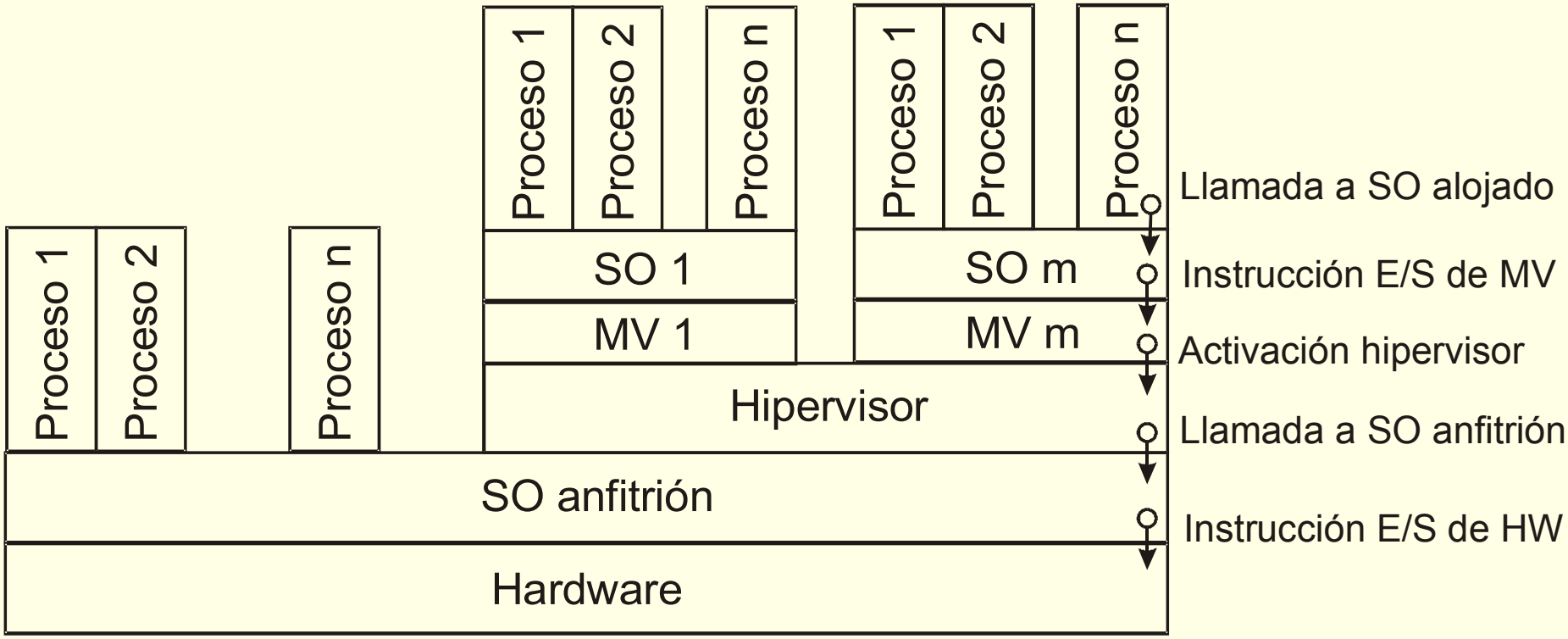
Máquinas virtuales de sistema

- Hipervisor crea MV multiplexando HW
  - SO sobre cada instancia de MV
    - Puede incluir emulador si distintos juegos de instrucciones
- 2 tipos:
  - Tipo I (MV nativa): Hipervisor (HP) ejecuta sobre HW
  - Tipo II (MV alojada): Hipervisor ejecuta sobre SO anfitrión
    - Hipervisor invierte la abstracción de SO anfitrión
      - P.ej. Interrupción real → Señal UNIX → Interrupción MV
- Paravirtualización: SO alojado modificado
  - Más eficiente pero menos compatible
- Posibles beneficios: Coexistencia de distintos SO, aprovechamiento HW, facilita despliegue de apps., depuración de nuevo SO, *sandboxing* de apps., apps. tipo *legacy*,...
- Alternativa a tipo II: virtualización a nivel SO (contenedores, ...)

MV de sistema tipo I (nativa)



MV de sistema tipo II (alojada)

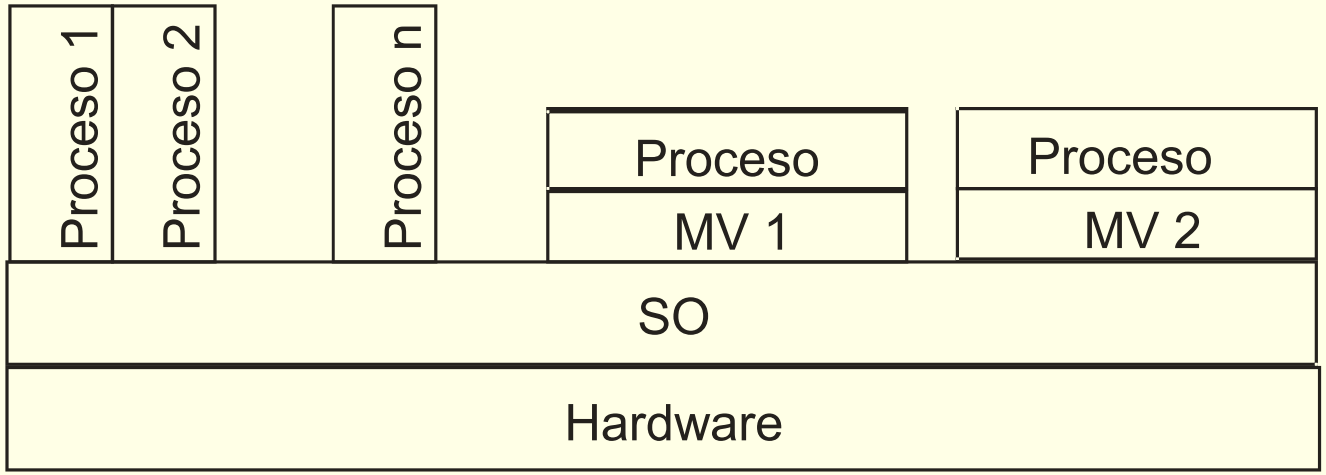


Implementación de MV de sistema

- Virtualización del procesador
  - Instrucciones no privilegiadas ejecutadas en UCP real
  - Instrucciones privilegiadas de SO alojado (SOA) → Hipervisor
  - ¿Cómo conseguirlo? Alternativas
    - SOA modo no privilegiado → excepción tratada por HP
    - Traducción binaria de instrucciones
    - Paravirtualización: SOA modificado llama directamente a HP
  - Instr. "sensibles" → privilegiadas (criterios Popek/Goldberg)
    - IA-32 no cumple (p.e. POPF) → Traducción binaria de instr.
  - Posibilidad de usar múltiples niveles de privilegio:
    - Nivel(Hipervisor) > Nivel(SO alojado) > Nivel(Aplicaciones)
- Virtualización de la memoria
  - Memoria física de SO alojado es espacio virtual creado por HP
    - 2 niveles de traducción: uso de tablas de página en la sombra

Máquinas virtuales de proceso

- Ejecuta como aplicación de SO y da soporte a un único proceso
- Proporciona juego de instrucciones ≠ real
- Puede corresponder con otro procesador pero su uso habitual:
  - Crear entorno ejecución independiente de HW y SO real
    - Ejecución de proceso aislada de otras aplicaciones (*sandbox*)
  - Ejemplos
    - *Java Virtual Machine*, *Common Language Runtime* de .NET



Principios de diseño del SO

- Los iremos descubriendo a lo largo de la asignatura.
- Anticipo:
  - SO debe definir mecanismos y no políticas
    - Ej. Generalmente, se da más prioridad a procesos con más E/S
      - No debería estar fijo en el SO, sino ser configurable
  - Portabilidad
    - SO escrito en lenguaje de alto nivel minimizando ensamblador
    - No siempre aprovechar toda la funcionalidad específica del HW
      - Ejemplo: SO que usa 4 niveles de privilegio de Pentium
  - Principio de mínimo privilegio
    - Software debe ejecutar sólo con privilegio que requiere
      - Ejemplo: demonios UNIX con permisos de superusuario