

Diseño de sistemas operativos

Presentación de la asignatura

Aspectos generales de la asignatura

- 9 créditos (4,5 teóricos + 4,5 prácticos)
- Horario de clases (4 horas/semana):
 - 41M (aula 3202):
 - Miércoles de 11 a 13 | Jueves de 9 a 11
- Clases presenciales en aula (≈ 60 h.), además de teoría, incluyen:
 - Presentación de prácticas y resolución de ejercicios
 - Tema de administración de sistemas
- Página web de la asignatura
 - <http://laurel.datsi.fi.upm.es/docencia/asignaturas/dso>
 - Noticias, foro, normas, publicación de material docente, ...
 - Consulta de notas

Evaluación de la asignatura

- Parte teórica + parte práctica
- Calificación:
 - $Total = 0,6 * Teoría + 0,4 * Prácticas + Optativas$
 - Mínimo compensable Teoría o Prácticas: 4,5 puntos
- Examen de teoría
 - Mantiene siempre misma estructura: 3 ejercicios
 - 1º ejercicio (4,5/10) vinculado con los temas:
 - ▶ Introducción, Procesos, Planificación, Interbloqueos y Memoria.
 - 2º ejercicio (4,5/10) vinculado con los temas:
 - ▶ Entrada/salida, Ficheros y Seguridad.
 - 3º ejercicio (1/10) vinculado con el tema:
 - ▶ Administración de sistemas
- Se considera como presentado a un alumno sólo si ha hecho:
 - El examen de teoría o el de prácticas

Prácticas

- Carácter no presencial
- Grupos de 2 (también individuales)
- 2 obligatorias:
 - *minikernel*: vinculada al tema de procesos
 - *memon*: vinculada al tema de gestión de memoria
 - Optativas: 1 punto adicional/cada una (sólo si $Total \geq 5$)
 - Basadas en el *minikernel*
 - Desarrollo de módulos en Linux
- Plazo de entrega único y común
 - 1 día antes de examen de la convocatoria
- Nota mínima compensable en cada práctica 4
- Examen conjunto de ambas prácticas de tipo test
 - Mismo día que examen de la parte teórica

Desarrollo de prácticas

- En sistema Linux
- Máquina del centro de cálculo: triqui.fi.upm.es
- Alumno puede usar su propia máquina pero entrega en **triqui**
- Ciclo de vida de la práctica:
 - Descarga de material de apoyo de página web de asignatura
 - Instalación material de apoyo
 - Desarrollo de (parte de) la funcionalidad pedida
 - Entrega de la práctica (sólo desde **triqui**)
 - Corrección automática (0, 13, 16 y 19 horas)
 - Resultado de corrección → correo cuenta del alumno en **triqui**
 - Número de entregas ilimitado
 - Última entrega se considera la versión definitiva

Profesores

- Teoría:
 - María de los Santos Pérez Hernández (mperez@fi.upm.es)
 - Pedro de Miguel Anasagasti (pmiguel@fi.upm.es)
 - José María Peña Sánchez (jmpena@fi.upm.es)
 - Francisco Rosales García (frosal@fi.upm.es)
 - Fernando Pérez Costoya (fperez@fi.upm.es) [coordinador]
- Prácticas:
 - Fernando Pérez Costoya (fperez@fi.upm.es)
 - Despacho 4201

Temario

- Introducción (**Jo**)
- Procesos (**Ma**)
 - Práctica 1: *minikernel* (**Fe**)
- Planificación (**Fe**)
- Interbloqueos (**Fe**)
- Gestión de memoria (**Fe**)
 - Práctica 2: *memon* (**Fe**)
- Entrada/salida (**Fr**)
- Sistemas de ficheros (**Pe**) **2011**
- Administración (**Jo**) **2012**
- Seguridad y protección (**Jo**)
- Introducción a los sistemas distribuidos (**Jo**)

Bibliografía

- *Sistemas Operativos: Una visión aplicada*. J. Carretero, P. de Miguel, F. García y F. Pérez. McGraw-Hill, 2007 (2ª Ed.)
 - Disponible en PDF para los alumnos de la asignatura
- *Operating Systems Concepts*. A. Silberschatz, P.B. Galvin, G. Gagne. John Wiley & Sons, 2008 (8ª Ed.)
- *Modern Operating Systems*. A.S. Tanenbaum. Prentice-Hall, 2007 (3ª Ed.)
- *Operating Systems: Design and Implementation* A.S. Tanenbaum, A.S. Woodhull. Prentice-Hall, 2006 (3ª Ed.)
- *Operating Systems* W. Stallings. Prentice-Hall, 2008 (6ª Ed.)

Diseño de sistemas operativos

Tema 1 *Introducción*

Contenido

- *Definición de SO*
- *Historia de los SSOO*
- *Componentes del SO*
- *Estructura del SO*
- *Principios de diseño del SO*
- *Administración de sistemas*

Encaje de la asignatura

- Precedentes:
 - SO (2º curso)
 - ¿**Qué** servicios ofrece un SO?
 - Objetivo: Construir aplicaciones que usan servicios del SO
 - Arquitectura (3º curso)
 - Sistemas de E/S y de memoria
- Diseño de sistemas operativos
 - ¿**Cómo** está diseñado internamente un SO?
 - Alternativas de diseño
 - Algoritmos y modo de operación internos
 - Además: aspectos básicos de administración

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 micrónúcleo
 - ▶ 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*)

Historia de los sistemas operativos

- Marcada por el desarrollo del hardware
- SO conservador: HW revolución pero SO evolución
- La historia es a veces cíclica
 - Técnicas que quedan obsoletas vuelven a recuperarse
 - “Ontogenia recapitula filogenia” (Tanenbaum)
- Distinguimos las etapas (aunque la realidad es continua):
 - Sistemas primitivos (≈1950)
 - Sistemas por lotes (≈1960)
 - Sistemas multiprogramados y de tiempo compartido (≈1970)
 - Sistemas basados en computadores personales (≈1980)
 - Situación actual

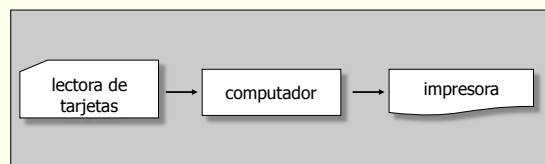
Sistemas primitivos

- Tecnología: tubos de vacío
 - Enorme coste, gran tamaño y consumo, poca fiabilidad.
- Uso absolutamente minoritario (científico y militar)
- Modo de operación interactivo:
 - Usuario es operador y programador
 - Programas deben gestionar todo el hardware
 - Usuario introduce programa mediante clavijas e interruptores
 - Obtiene resultados en *leds*
 - Posteriormente, uso de tarjetas perforadas y cinta de papel
- Uso ineficiente de recursos
- Sin software de sistema
 - Aunque empiezan a surgir bibliotecas de subrutinas comunes

Sistemas por lotes

- Tecnología: transistores (no integrados)
 - Grandes mejoras en coste, volumen, consumo y fiabilidad
- Uso comercial en empresas muy grandes
 - Gran inversión: hay que sacarle partido
- Modo de operación por lotes (*batch*) no interactivo:
 - Se distingue operador de programador
 - Operador agrupa trabajos similares en bandejas de tarjetas
 - Trabajo: tarjetas de control (p.e. qué lenguaje) + código + datos
 - Computador ejecuta cada trabajo imprimiendo sus resultados
- Software de sistema
 - Compiladores, ensambladores, ...
 - Primeros s. operativos (GM-NAA I/O para IBM 704, 1956)
 - Monitor residente: secuenciador automático de trabajos

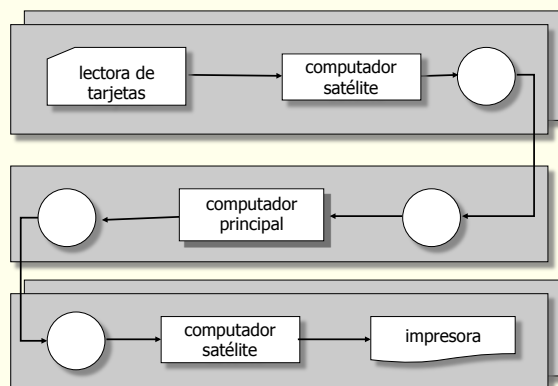
Sistemas por lotes



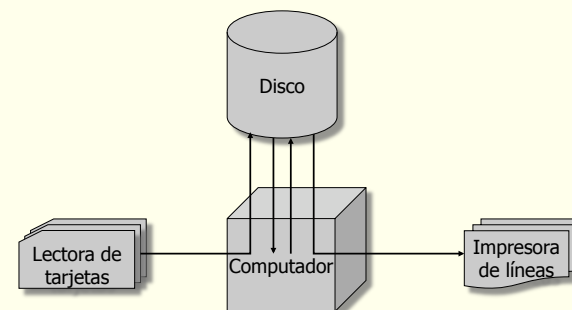
Sistemas por lotes *offline y spooling*

- Rendimiento de sist. por lotes acotado por lectora e impresora
- Aparición de cintas magnéticas
- Computador lee trabajos y escribe resultados en cintas
 - Computador(es) satélite(s) (de muy bajas prestaciones):
 - Leen tarjetas a cintas e imprimen resultados desde cintas
 - Operadores transportan cintas entre computadores
- Aparición de discos y mejora en técnicas de E/S (DMA)
 - **Simultaneous Peripheral Operation On Line (SPOOL)**
 - Desaparece la necesidad de computadores satélites
 - Mientras se ejecuta trabajo:
 - Se lee siguiente de tarjetas a disco
 - Se imprimen resultados del anterior desde el disco

Procesamiento *offline*



Spooling



Sistemas por lotes multiprogramados

- Tecnología: circuitos integrados
- Uso en todo tipo de organizaciones
- Mayor desfase entre prestaciones procesador/memoria y E/S
 - Procesador casi siempre parado
 - Capacidad de memoria permite cargar múltiples programas
- **Multiprogramación**
 - Se reparte el uso del hardware entre programas activos
 - Cuando programa espera E/S, SO cede control a otro
 - Esta técnica dispara la complejidad del SO
- Surgen los *mainframes*
 - Familia de sistemas OS/360 de IBM
 - SO que llego a tener enorme complejidad
 - *The Mythical Man-Month* (Brooks)

Sistemas de tiempo compartido

- Trabajo por lotes: baja productividad en programación
 - Se necesita interactividad
- **Tiempo compartido**
 - Usuario trabaja directamente con máquina mediante terminal
 - Cree que tiene una máquina propia
 - SO reparte equitativamente recursos entre usuarios
- Algunos sistemas notables:
 - CTSS: primer sistema de tiempo compartido de entidad
 - MULTICS: ambicioso proyecto frustrado pero muy relevante
 - UNIX: escrito en C en vez de en ensamblador
- Empresas informáticas verticales: HW, SO y aplicaciones
- Tres tipos de computadores con su tipo de SO específico
 - *Mainframes*, minicomputadores (VAX) y supercomputadores

Sistemas basados en computadores personales

- Tecnología: (V)LSI
- Computadores personales: informática ubicua
 - Necesidad de una interfaz de usuario muy sencilla (GUI)
 - Alta interactividad
- Uso monousuario pero con necesidad de multiprogramación
- Primeros muy rudimentarios y monoprogramación (MS-DOS)
- Posteriores (Windows NT, Linux) similares a SO convencionales
 - Tendencia a usar mismo tipo de SO para distintas plataformas
- Tecnología: Desarrollo espectacular de redes de comunicación
 - Incorporación de software de comunicación en SO
 - Pionero UNIX 4.2 BSD (1983)
 - Surgen los sistemas operativos distribuidos (otra asignatura)

Situación actual

- Gran variedad de plataformas:
 - Desde portátiles y sistemas empotrados a supercomputadores
 - Tendencia: mismo SO apropiado para todo este rango
- Proliferación de multiprocesadores con distintos niveles
 - *Hyperthreading*, multinúcleo, sistemas UMA y NUMA
 - SO debe aprovechar al máximo todo ese paralelismo
- Sistemas de tiempo real
 - Críticos: requieren SO específico RTOS
 - No críticos: SO convencionales van adaptándose
 - P.ej. Linux 2.6 ha pasado a ser un núcleo expulsivo
 - ▶ Proceso puede ser expulsado en medio de una llamada al sistema
 - Sistemas multimedia caen dentro de esta categoría

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

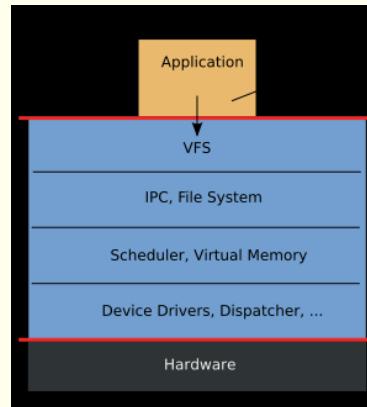
Estructura del sistema operativo

- 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)



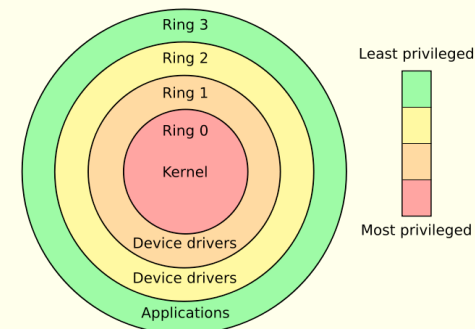
Sistemas con módulos cargables

- 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 empujado
 - 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)



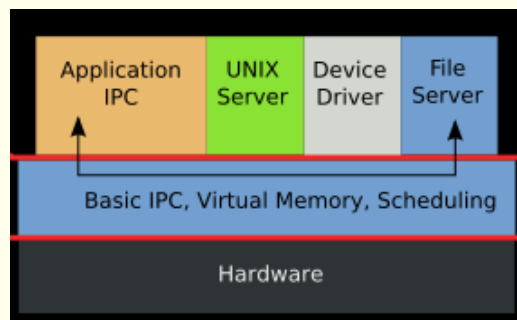
Micronúcleo (I)

- “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.

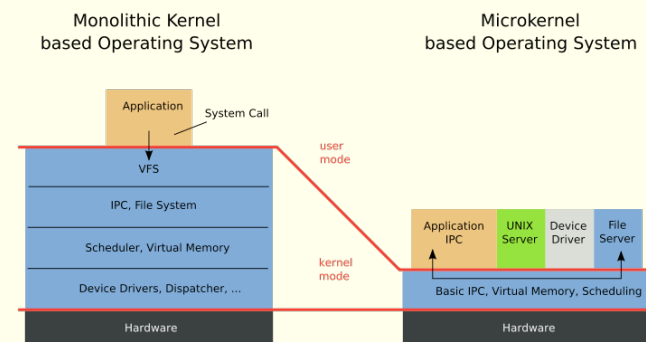
Micronúcleo (II)

- 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 \approx 100 llamadas; L4 \approx 10 llamadas
 - Nanonúcleos, piconúcleos (actualmente = micronúcleos)
- 3ª generación: núcleos verificados formalmente (SeL4)

Sistema basado en micronúcleo (wikipedia)



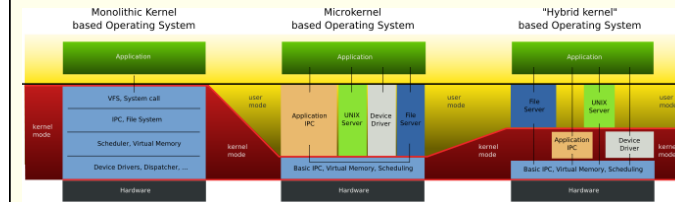
Monolítico versus micronúcleo (wikipedia)



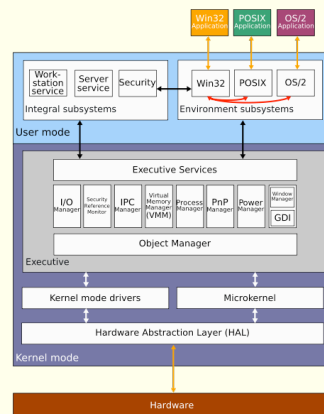
Sistemas híbridos

- 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)



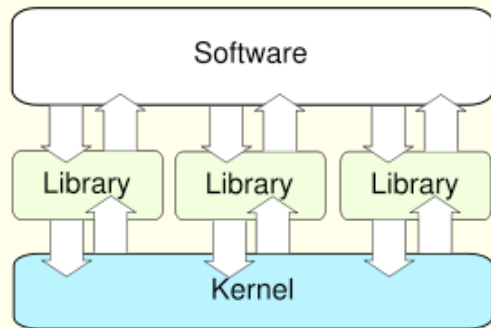
Estructura de Windows 2000 (wikipedia)



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

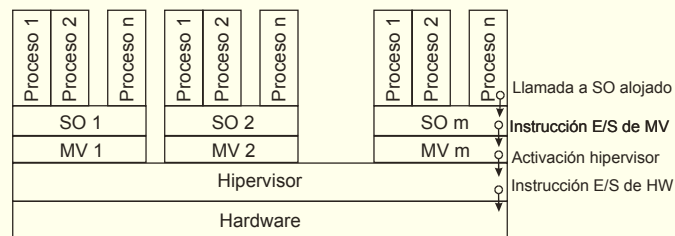
Tipos de máquinas virtuales

- 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

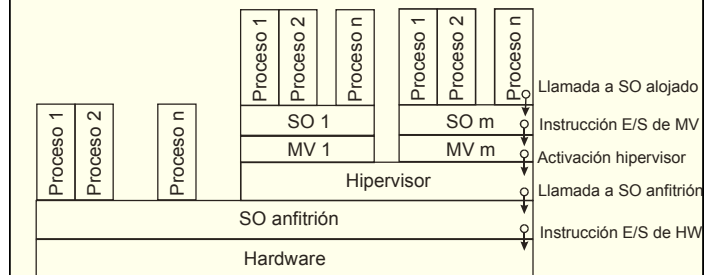
Máquinas virtuales de sistema

- 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
 - Depuración de nuevo SO, disponibilidad,...
- Alternativa a tipo II: virtualización a nivel SO (contenedores, ...)

MV de sistema tipo I (nativa)



MV de sistema tipo II (alojada)

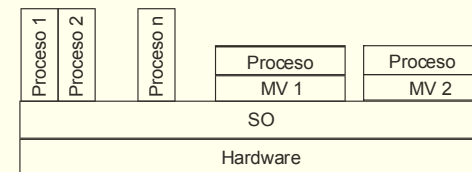


Implementación de MV de sistema

- 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 de 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



Ejemplo: User Mode Linux (UML)

- Simula “linux dentro de linux”
- Orientado a la depuración de versiones de kernel o módulos.
- El Linux alojado corre como un proceso
 - Define un “soporte” a una arquitectura linux del kernel.
- El procesos que usan el linux alojado son procesos del linux anfitrión
 - Pero redirigen sus llamadas al linux alojado



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

Administración de sistemas

- Fases del proceso de administración
 - Selección del SO, de la “distribución” y del soporte
 - Instalación del sistema
 - Selección del tipo de instalación
 - Configuración básica de contexto, particiones, red, paquetes, ...
 - Configuración posterior del sistema
 - Usuarios, sistemas de ficheros, red, dispositivos, servidores, ...
 - Instalación de nuevo software y hardware
 - Implantación de políticas de seguridad ante contingencias
 - Monitorización y ajuste del sistema
 - Automatización de tareas de administración (*scripting*)

Generación del SO

- Soporte de instalación incluye imágenes del SO precompiladas
- Administrador puede querer instalar una nueva versión
- Fases (suponiendo que se dispone de fuente de nueva versión)
 - Configuración (Linux: *make config*).
 - Ejemplos de parámetros configurables:
 - ▶ Código en imagen vs. en módulo, soporte SMP, núcleo expulsivo, ...
 - Selecciona fuentes y fija valores para compilación condicional
 - Compilación/construcción de la imagen (Linux: *make bzImage*)
 - Y de sus módulos (Linux: *make modules*)
 - Instalación de la nueva imagen y módulos
 - Copiar a directorios convenientes (Linux: *make modules_install*)
 - Actualizar configuración de cargadores de *boot* para usar imagen
 - ▶ GRUB de GNU, LILO de Linux