

Estructura del Sistema Operativo

Módulo 2

Departamento de Informática
Facultad de Ingeniería
Universidad Nacional de la Patagonia "San Juan Bosco"

Estructuras de Sistemas Operativos

- Servicios de Sistemas operativos
- Interfaz de Usuario del Sistema Operativo
- Llamadas a Sistema
- Tipos de Llamadas a Sistema
- Programas de Sistemas
- Diseño e Implementación de un Sistema Operativo
- Estructura de un Sistema Operativo
- Máquinas Virtuales
- Depuración de un Sistema Operativo
- Generación de un Sistema Operativo
- Boot del Sistema

Objetivos

- Describir los servicios que ofrece un sistema operativo a usuarios, procesos y otros sistemas.
- Discutir las distintas formas de estructurar un sistema operativo.
- Explicar como son instalados los sistemas operativos, personalizados y como se inician.

Servicios del Sistema Operativo

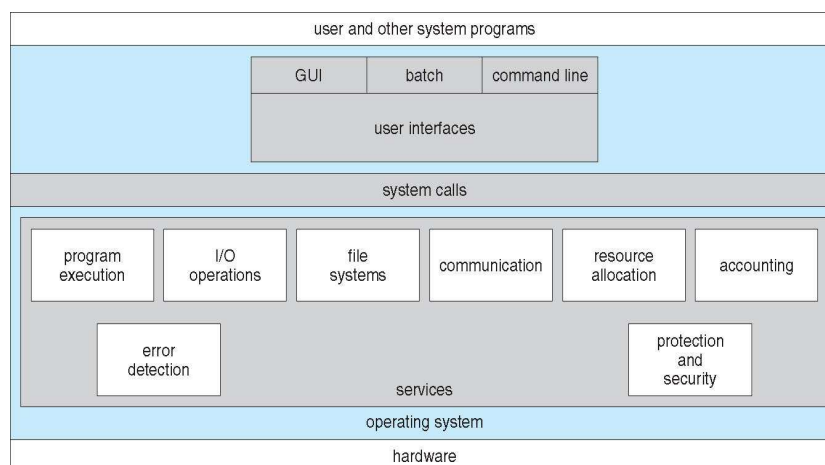
- Podemos ver un SO desde varios puntos de vista.
- Centrados en:
 1. **Servicios** → dirigido a programadores
 2. **interfaz** y los **programas** → a usuarios
 3. **componentes** y sus **interconexiones** → a diseñadores

Servicios del Sistema Operativo

Funciones del SO útiles a los usuarios:

- **Interfaz de Usuario** – (CLI), **Graphics User Interface** (GUI), Batch
- **Ejecución de Programas**- Carga en memoria, ejecución y terminación en forma normal o anormal (indicando el error).
- **Operaciones de E/S** - dirigida a un archivo o a un dispositivo de E/S.
- **Manipulación del Sistema de Archivos** - Operaciones sobre archivos y directorios para: leer, escribir, crear, borrar, buscar usando su nombre, listar información y la administración de permisos.
- **Comunicaciones**– entre procesos de la misma computadora o a través de la red. (memoria compartida o paso de mensajes).
- **Detección de errores**– en el hardware, la memoria, en dispositivos de E/S, en programas de usuario.
- Para c/tipo de error el SO debe ejecutar la operación apropiada para asegurar una computación correcta y consistente

Una Visión de los Servicios de un Sistema Operativo



Servicios del Sistema Operativo (Cont)

Existe otro conjunto de funciones del SO para **garantizar la eficiencia** del propio sistema a través de recursos compartidos.

- **Asignación de Recursos:** con múltiples usuarios o tareas ejecutándose al mismo tiempo, deben ser asignados a cada uno, los recursos necesarios.
- **Contabilidad** – Lleva la cuenta de, que usuarios emplean qué recursos y en qué cantidad.
- **Protección y seguridad** – En la ejecución de procesos concurrente, no debe ocurrir que un proceso interfiera con los demás o con el propio SO.

Servicios del Sistema Operativo (Cont)

- **Protección** implica asegurar que todos los accesos a recursos del sistema estén controlados.
- **Seguridad:** El sistema requiere autenticación de cada usuario mediante contraseña, para tener acceso a los recursos del sistema. La seguridad se extiende en defensa de los dispositivos externos de E/S (modems, red), frente a intentos de accesos no válidos.

Para que un sistema esté protegido y seguro, la protección deben implementarse en todas sus partes pues **una cadena es tan fuerte como el más débil de los eslabones**.

Interfaz de Usuario del Sistema Operativo - CLI

El **intérprete de comando** o (Command Line Interface -CLI) permite ingresar comandos en forma directa

Función principal: **obtener** y **ejecutar** el siguiente comando.

Implementado de dos formas:

- el **intérprete** contenga el código de cada comando.
- ó como **programas del sistema**. (SO mas flexible a cambios y una CLI mas pequeña.)

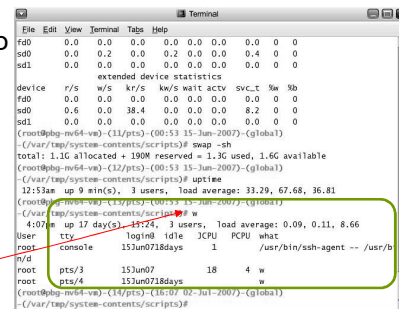
Agregar nuevas características, no requiere modificar la shell.

Puede tener múltiples variantes **shells**

► Ej comando

rm file.txt

w



The screenshot shows a terminal window with the following content:

```
File Edit View Terminal Tabs Help
fd0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
sd0 0.0 0.2 0.0 0.2 0.0 0.0 0.4 0.0 0.0 0.0 0.0
sd1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
extended device statistics
device r/s w/s kr/s kw/s wait actv svc_t %w %b
fd0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
sd0 0.6 0.0 38.4 0.0 0.0 0.0 8.2 0.0 0.0 0.0 0.0
sd1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
(root@hqp-m64-va)-(11/pts)-(00:53 15-Jun-2007)-(global)
-/var/tmp/system-contents/scripts# swap -sh
total: 1.1G allocated + 180M reserved = 1.3G used, 1.6G available
-/var/tmp/system-contents/scripts# uptime
12:13am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
-/var/tmp/system-contents/scripts# w
4:07pm up 17 day(s)-49:24, 3 users, load average: 0.09, 0.11, 8.66
USER TTY LOGGED IDLE CPU PCPU WHAT
root console 15Jun07 18days 1 /usr/bin/ssh-agent -- /usr/b
n/d pts/3 15Jun07 18 4 w
root pts/4 15Jun07 18 w
-/var/tmp/system-contents/scripts#
```

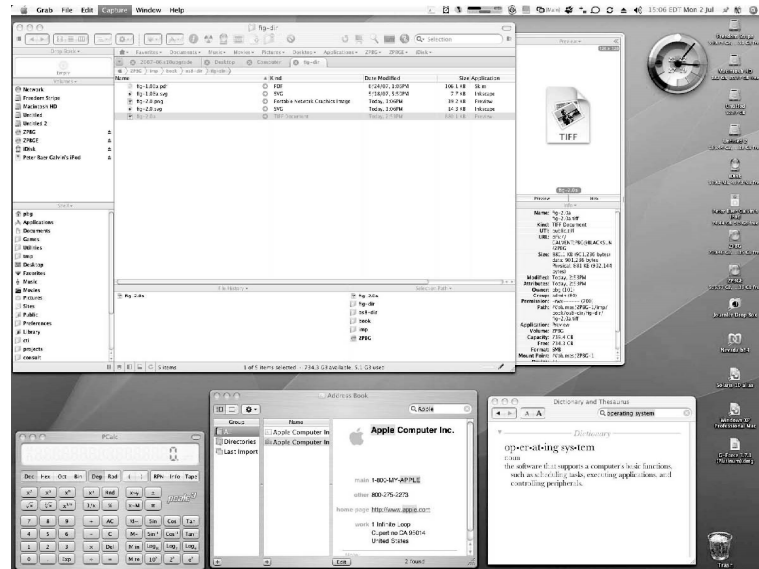
Interfaz Gráfica de Usuario del SO - GUI

Interfaz **desktop** “amigable”

- Emplea sistema de ventanas y menú controlados con **mouse**.
- Los **iconos** representan archivos, programas, acciones, etc.
- Los botones del mouse sobre objetos causan acciones (mostrar información, desplegar menú, ejecutar funciones, abrir directorios (**carpetas** o **folder**))
- Inventada por Xerox PARC '70
- Casi todos los sistemas **incluyen** interfaces CLI y GUI
 - Microsoft Windows basada en GUI con una CLI que es una Shell.
 - Apple Mac OS X tiene interfaz GUI “Aqua” soportada por un kernel UNIX por debajo y sus shells disponibles.
 - Solaris es CLI con una interfaz GUI opcional (Java Desktop, KDE)



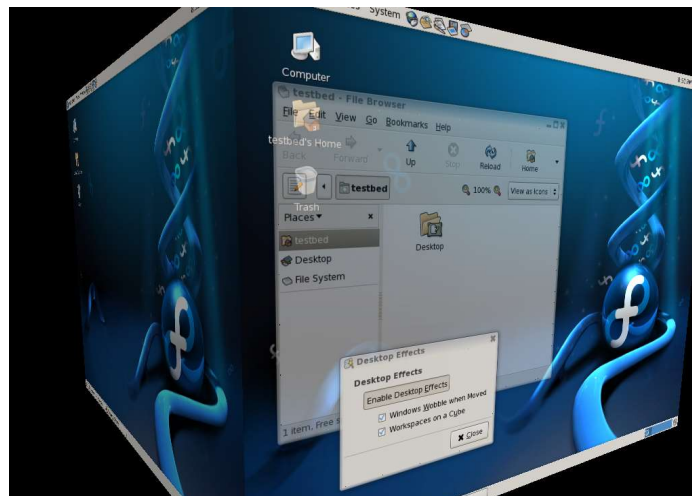
La GUI Mac OS X



JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Tendencias actual: Interfaz de usuario 3D



Ejemplo de software con interfaz de usuario en 3D Compiz de Novell y AIGLX corriendo en Fedora Core 6

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

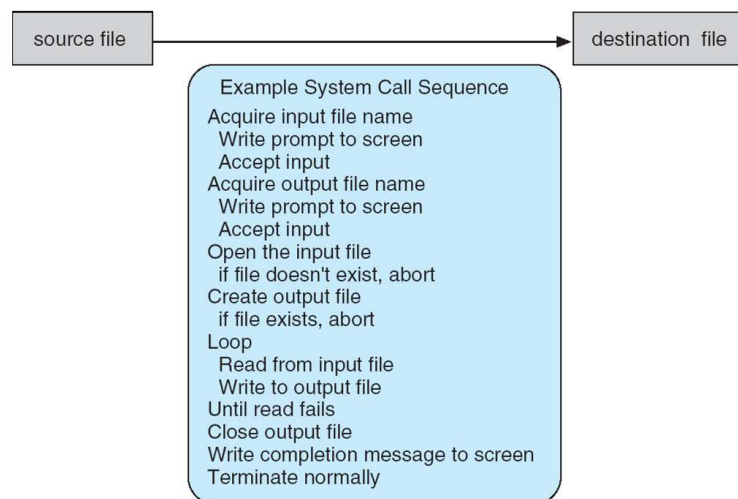
Llamadas al Sistema

- Son la interfaz de programación a los servicios provistos por el SO
- Típicamente escritas en lenguajes de alto nivel (C o C++)
- En general son accedidas por programas vía (API) [Application Program Interface](#), más que por el uso directo a llamadas a sistema
- Las tres API más comunes son
 - ▶ **API Win32** para sistemas Windows,
 - ▶ **API POSIX** para sistemas POSIX (incluye todas las versiones de UNIX, Linux, y Mac OS X), y
 - ▶ **API Java** para una máquina virtual Java (JVM)

POSIX "Portable Operating System Interface [Unix] familia de normas relacionadas con la especificada por la IEEE para definir APIs,

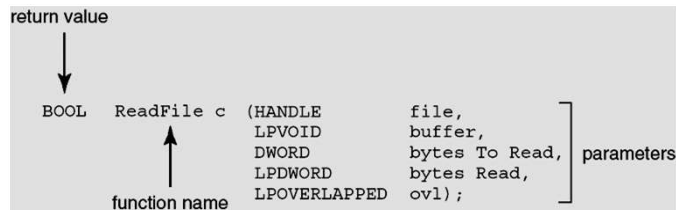
Ejemplo de Llamada a Sistema

- Llamada a sistema para copiar el contenido de un archivo a otro.



Ejemplo de una API Standard

- Considere la función **ReadFile()** en la Win32 API (para leer de un archivo)

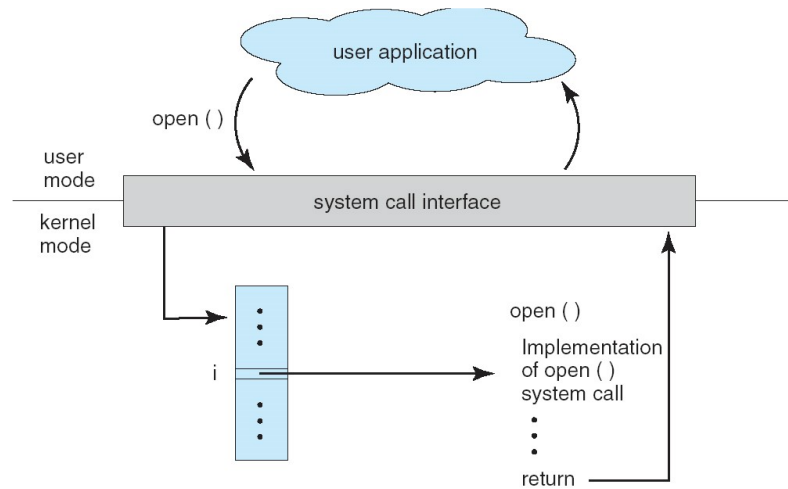


- Parámetros pasados a **ReadFile()**
 - HANDLE** archivo—el archivo a ser leído
 - LPVOID** buffer—dirección del buffer que recibirá el dato y se leerá
 - DWORD** bytes a leer— el número de bytes a leer del buffer
 - LPDWORD** bytes leídos— el número de bytes leídos durante la última lectura
 - LPOVERLAPPED** solapada—indica si permite E/S **asíncrona** es decir que el proceso continúe mientras se produce la E/S

Implementación de Llamadas a Sistema

- Cada llamada al sistema tiene asociado un **número**.
 - La API mantiene una **tabla indexada** de acuerdo con estos números
- La interfaz invoca la llamada necesaria del kernel del SO y devuelve el **estado** de la llamada a sistema y los posibles valores.
- Quien realiza la llamada no tiene que saber como está implementada.
 - Solo necesita invocar la API y entender lo que hará el SO como resultado de la llamada
 - La API oculta al programador los detalles de la interfaz del SO.
 - Éstos son manejados por librerías run-time (funciones incluidas con el compilador)

Relación API – Llamada a Sistema del SO

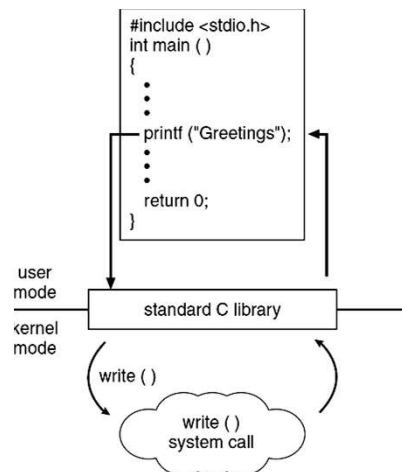


JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Ejemplo de Librería Standard C

- Un programa C invoca la llamada de librería `printf()`, la cual invoca a la llamada a sistema `write()`

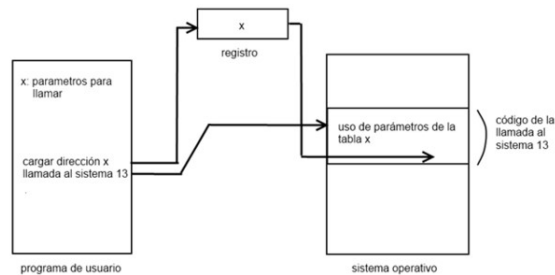


JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Pasaje de Parámetros en Llamadas a Sistema

- Un System Call en general requiere más información que sólo el identificador. El tipo y la cantidad de información varía de acuerdo al SO
- Existen tres métodos para pasar parámetros al SO
 - *en **registros**, (si la cantidad parámetros < registros)
 - *en un **bloque**, o tabla, en memoria. La dirección del bloque es pasada como parámetro en un registro. (en Linux y Solaris)



*en una **pila**. Puestos por el programa y quitados de la pila por el SO. Los métodos por bloque y pila no limitan el número de parámetros a ser pasados

Tipos Llamadas a Sistema

- Control de procesos
- Administración de archivos
- Administración de dispositivos
- Mantenimiento de Información
- Comunicaciones
- Protección

Tipos Llamadas a Sistema

- Control de procesos
 - *terminar, abortar*
 - *cargar, ejecutar*
 - *crear procesos, terminar procesos*
 - *obtener atributos del procesos, definir atributos del proceso*
 - *esperar para obtener tiempo*
 - *esperar suceso, señalar suceso*
 - *asignar y liberar memoria*
- Administración de archivos
 - *crear archivos, borrar archivos*
 - *abrir, cerrar*
 - *leer, escribir, reposicionar*
 - *obtener atributos de archivo, definir atributos de archivo*

Tipos Llamadas a Sistema

- Administración de dispositivos
 - *solicitar dispositivo, liberar dispositivo*
 - *leer, escribir, reposicionar*
 - *obtener atributos de dispositivo, definir atributos de dispositivo*
 - *conectar y desconectar dispositivos lógicamente*
- Mantenimiento de Información
 - *obtener la hora o la fecha, definir la hora o la fecha*
 - *obtener datos del sistema, establecer datos del sistema*
 - *obtener atributos de procesos, archivos o dispositivos*
 - *establecer los atributos de procesos, archivos o dispositivos*

Tipos Llamadas a Sistema

- Comunicaciones
 - *crear, eliminar conexiones de comunicación*
 - *enviar, recibir mensajes*
 - *transferir información de estado*
 - *conectar y desconectar dispositivos remotos*

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Programas de Sistema

- Los programas de sistema proveen un medio para el **desarrollo** y la **ejecución** de programas. Pueden ser divididos en:
 - ◆ Manipulación de archivos
 - ◆ Información de estado
 - ◆ Modificación de archivos
 - ◆ Soporte de lenguajes de programación
 - ◆ Carga y ejecución de programas
 - ◆ Comunicaciones
 - ◆ Programas de aplicación

La visión que tienen la mayoría de los usuarios del SO está dada por los **programas de sistema** y no por las **llamadas a sistema** (system calls).

Programas de Sistema

- Administración de archivos – programas que manipulan archivos y directorios (Crea, borran, copian, renombran, imprimen, vuelcos, listan, tanto archivos como directorios)
- Información de estado Programas que solicitan al sistema – (fecha, hora, cantidad de memoria disponible, espacio de disco, número de usuarios)
 - Otros proveen detalles de rendimiento, bitácoras y mecanismos de depuración, inicio de sesión
 - Los datos de salida se envían a terminales, archivos, o a otros dispositivos de salida x ejemplo impresora o ventana en la GUI.
- Modificación de archivos Puede disponer de varios editores de texto para crear y modificar archivos

Programas de Sistema (cont)

- Soporte de lenguajes de programación - Compiladores, ensambladores, depuradores e intérpretes
- Carga y ejecución de programas – Cargadores absolutos, cargadores reubicables, editores de enlace, y cargadores de overlay (sustitución), sistemas depuradores para lenguajes máquina y alto nivel
- Comunicaciones – Proveen mecanismos para crear conexiones virtuales entre procesos, usuarios y sistemas de cómputo
 - Permite a los usuarios enviar mensajes a pantallas o correo electrónico, navegar páginas web, conectarse remotamente, transferir archivos de una máquina a otra.

Además de los programas del sistema, los SO cuentan con programas **utilitarios** para resolver problemas comunes como: explorador web, editores, sistema de base de datos, juegos etc llamados **Programas de aplicación**

Diseño e Implementación de un SO

¿A que nos enfrentamos al diseñar e implementar un SO?

- La estructura interna de diferentes SOs puede variar ampliamente
- Inicialmente hay que definir los objetivos y las especificaciones
- Se deberá elegir el hardware, tipo de sistema (p x lotes, t. compartido, monousuario, multiusuario, distribuido, en t. real o de propósito general)
- Se deberá especificar los requisitos:
 1. **Objetivos del Usuario** – El SO debería ser adecuado para su uso, fácil de aprender, confiable, seguro y rápido
 2. **Objetivos del Sistema** – El SO debería ser fácil de diseñar, implementar y mantener, también flexible, confiable, libre de errores y eficiente.

Diseño e Implementación de un SO

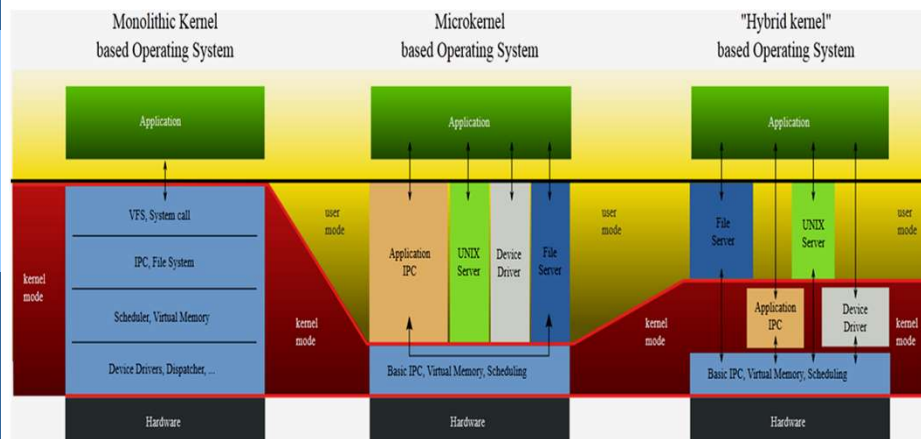
- Un principio importante es la separación
 - Política:** ¿Qué deberá hacerse?
 - Mecanismo:** ¿Cómo hacerlo?
- Los mecanismos determinan como hacer algo, las políticas deciden que debe hacerse
 - La separación permite máxima flexibilidad.
 - Las decisiones políticas pueden cambiar con el paso del tiempo.
 - Sería deseable que un mecanismo no se vea afectado por cambios de política.
- Los SO basados en **microkernel** llevan al extremo esta separación, implementando un conjunto **básico** de componentes primitivos independiente de las políticas. Permite agregar políticas y mecanismos avanzados en módulos del *kernel*.

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Estructura del SO

- Para desarrollar un SO moderno es importante dividir la tarea en componentes más pequeños, en lugar de tener un sistema monolítico.

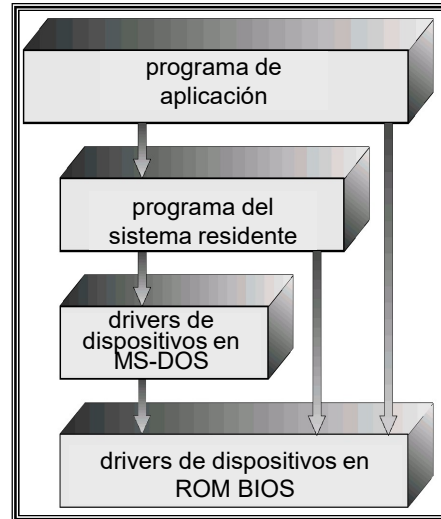


JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Estructura Simple: Capas de MS-DOS

- MS-DOS – escrito para proveer máxima funcionalidad en el menor espacio
- No fue dividido en módulos
- Aunque MS-DOS tiene cierta estructura, sus interfaces y niveles de funcionalidad no están bien separados
- Los programas de aplicación pueden acceder a las rutinas básicas de E/S (BIOS) para escribir en pantalla y en disco.
- No tenía modo dual ni protección HW.

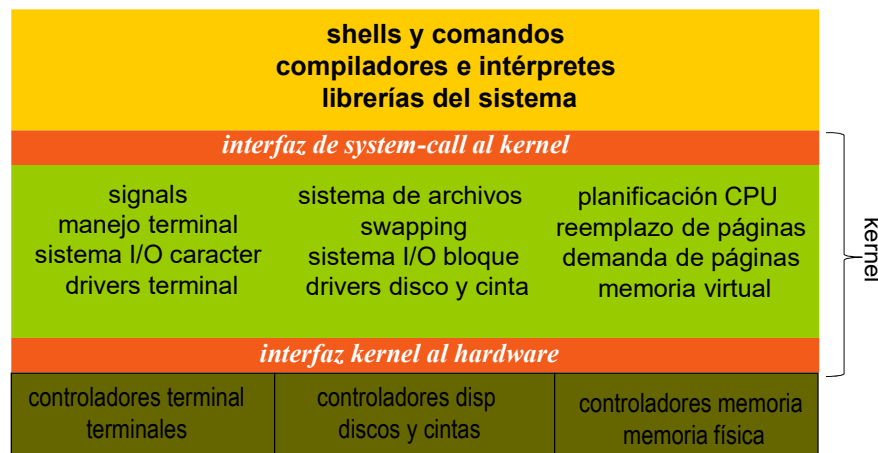


JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Estructura de UNIX

USUARIOS



JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

UNIX

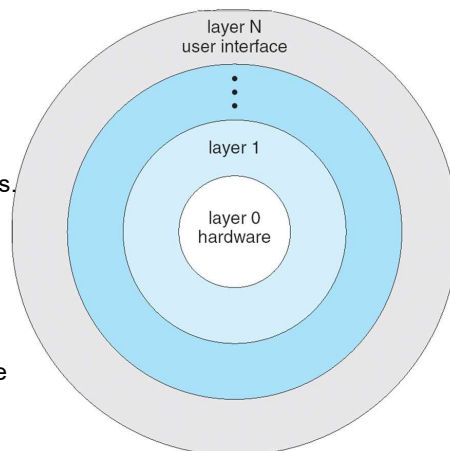
- UNIX original – estaba limitado por la funcionalidad del hardware.
- Consta de dos partes separadas.
 - Programas de sistema
 - El kernel
 - ▶ Comprende todo lo que está debajo de la interfaz de los system calls y encima del hardware
 - ▶ Contiene el sistema de archivos, la planificación de CPU, manejo de memoria, y otras funciones del sistema operativo;
 - ▶ Es una enorme cantidad de funcionalidad en un solo nivel.

Enfoque por Capas

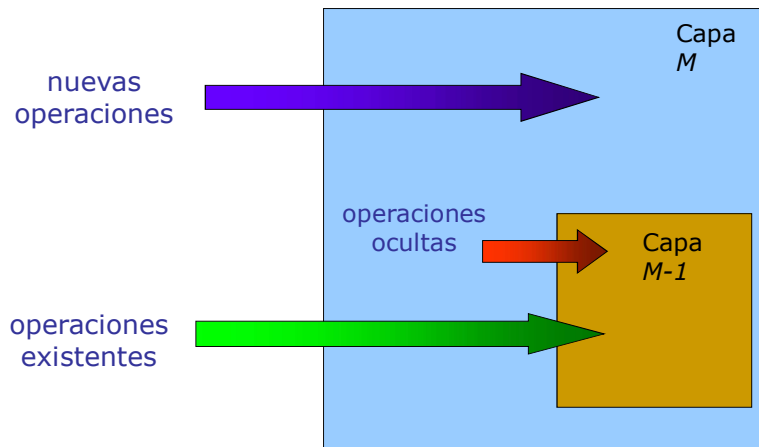
- SO dividido en capas (niveles), cada una construida sobre el tope de otra. La capa inferior (nivel 0), es el hardware; la mas alta (capa N) es la interfaz de usuario.
- Un nivel típico M consta de estructuras de datos y operaciones que los niveles superiores pueden invocar.
- Cada capa usa funciones y servicios de las capas inferiores.

Dificultad → definir los diferentes niveles apropiadamente.

Las llamadas tardan mas en ejecutarse porque agrega carga de trabajo adicional



Sistema Operativo por Capas



JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Estructura de Sistema Microkernel

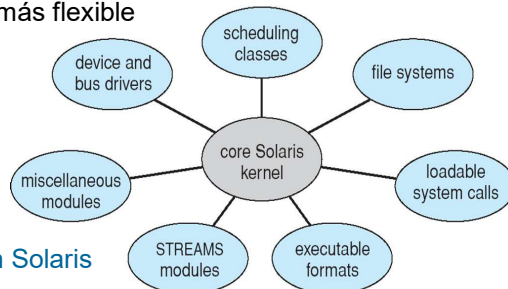
- Elimina todos los componentes no esenciales del kernel y los implementa como programas del sistema en el espacio de usuario:
→ Kernel más pequeño. (SO Mach)
- Las comunicaciones tienen lugar entre módulos de usuarios por medio de pasajes de mensajes
- Beneficios:
 - Más fácil de extender
 - Más fácil de portar el SO a nuevas arquitecturas
 - Mas confiable (menos código corre en el modo kernel)
 - Más seguro (si falla un servicio, el resto del SO no se ve afectado)
- Detrimentos:
 - Sobrecarga de rendimiento en la comunicación del espacio de usuario al espacio de kernel

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Módulos

- Los SOs más modernos implementan el kernel en *módulos*
 - Usa un enfoque orientado a objetos
 - Cada componente está separado del núcleo
 - Los protocolos de comunicación entre ellos son sobre interfaces conocidas
 - Cada módulo es cargado a la medida que se necesite dentro del kernel
- Similar a capas pero más flexible

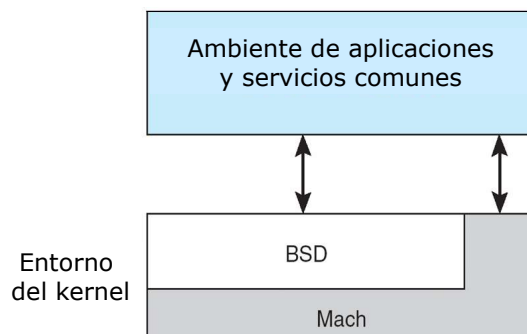


Enfoque Modular en Solaris

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Estructura híbrida de Mac OS X



Usa técnica por niveles

- microkernel Mach** (gestión Mem, RPC: llamadas a procedimiento remoto, IPC: comunicación entre procesos, paso msj, Planif. de hilos)
- kernel BSD** (CLI, soporte para red y Sistema de Archivos, API de POSIX, Pthreads).
- Extensiones del kernel (módulos dinámicamente cargables)

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Máquinas Virtuales

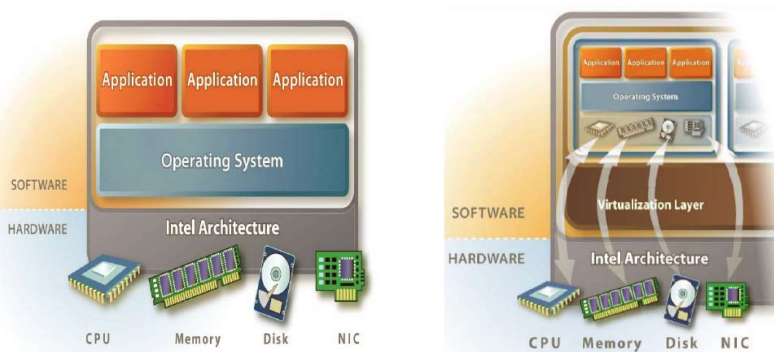
- Una *máquina virtual* se basa en el diseño por capas proyectado de manera lógica.
- Trata el *hardware* (CPU, memoria, unidades de disco, tarjeta de red, etc) y el *kernel* del SO como si fuera todo hardware.
- Una máquina virtual provee una interfaz *idéntica* al hardware primitivo subyacente.
- El SO forma varios entornos de ejecución diferentes, creando la ilusión de que cada entorno está operando en su propio procesador con su propia memoria (virtual).
- Cada *invitado* (guest) es provisto con una copia (virtual) de la computadora
- El software de la máquina virtual se ocupa de multiprogramar las múltiples máquinas virtuales sobre una única máquina física.
- Aparecieron comercialmente en las mainframes de IBM en 1972

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Beneficios de las Máquinas Virtuales (MV)

- Múltiples ambientes de ejecución concurrente: diferentes SOs pueden compartir el mismo hardware
- Cada MV está completamente aislada y protegida unos de otros.
- Se puede compartir archivos con *minidiscos*.
- La red comunicación virtual se implementa por software.
- Útil para investigación, desarrollo y testing de SO.



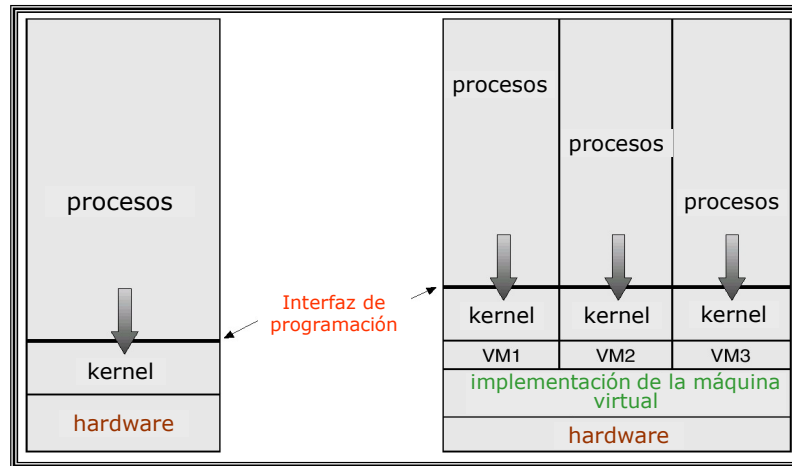
JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Modelos de Sistema

Máquina no virtual

Máquina virtual



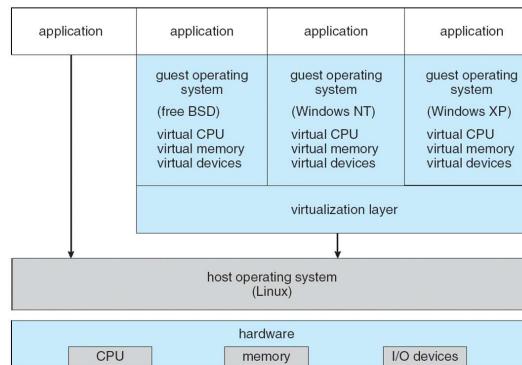
JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Arquitectura de VMware

VMware es una aplicación comercial que abstrae el HW 80x86 de Intel, creando una serie de máquinas virtuales aisladas.

Se ejecuta como una aplicación sobre un SO host, como Windows o Linux, y permite al sistema **host** ejecutar de forma concurrente varios SO huésped (**guest**) diferentes como máquina virtuales independientes.

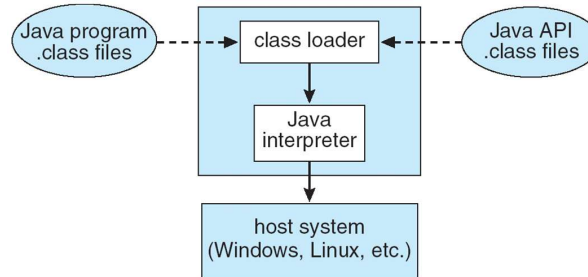


La capa de virtualización es el corazón de VMware, ya que abstrae el HW físico, creando máquinas aisladas que se ejecutan como SO huésped. Cada Maq virtual tiene su propia CPU, memoria, unidad de disco, interfaces de red, etc., virtuales.

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

La Máquina Virtual Java



El Lenguaje de programación OO Java, además una amplia biblioteca de APIs, provee una especificación para una JVM (Java Virtual Machine).

Cada programa consta de una o mas clases y el compilador Java genera un archivo (.class) en **código intermedio (bytecode)**, que se ejecutará sobre cualquier implementación de la JVM.

JVM es una especificación de una computadora abstracta. Consta de un **cargador de clases** y un **interprete Java** que ejecuta el código intermedio.

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Generación de Sistemas (SYSGEN)

- Los SO son diseñados para ejecutar en cualquier clase de máquina, por ello, el sistema debe ser configurado o generado para cada sitio específico de computación.
- El programa **SYSGEN** obtiene información concerniente a la configuración específica del hardware.
 - Que CPU va a usar, opciones instaladas
 - Cantidad de memoria disponible
 - Dispositivos instalados y sus características relevantes.
 - Buffer, tipo de planificador, nro procesos, etc
- Obtenida esta información, la descripción puede dar lugar a la creación de tablas y a la selección de módulos de una biblioteca precompilada para formar el Sistema Operativo final.

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Boot del Sistema

- El SO se debe poner a disposición del hardware para su uso, así el hardware puede iniciarlo
- *Booting* o arranque – inicio de la computadora mediante la carga del kernel.
- *Programa Bootstrap* – código almacenado en ROM que localiza el kernel, lo carga en la memoria e inicia su ejecución.
 - Con el encendido se inicializa el sistema, comienza la ejecución a partir de una dirección fija de memoria
 - El código inicial de boot se almacena en firmware y el SO en disco.
 - El proceso es en dos pasos: donde una parte pequeña del código accede al **boot block** en una locación fija del disco y carga el **bootstrap loader**
 - cargado el programa de arranque completo, puede localizar el kernel y cargarlo a memoria

JRA © - LRS 2025

Sistemas Operativos – Estructura de Sistemas Operativos

Fin del Módulo 2

Módulo 2

Departamento de Informática
Facultad de Ingeniería
Universidad Nacional de la Patagonia "San Juan Bosco"